November 2, 1978, David Kahn informed me of two associated conferences to be held in Bonn-Bad Godesburg and in Stuttgart, in November 1978, entitled "Welche Rolle spielte die Funkaufklärung für den Verlauf des Zweiten Weltkrieges?" and "Moderne Technologien und ihre Konsequenzen für die Kreigführung: Das Beispiel der Funkaufklärung."

REFERENCES

Beesly, Patrick. *Very Special Intelligence: the Story of the Admiralty's Operational Intelligence Centre.* London: Hamish Hamilton, 1977.

Brown, Anthony Cave. *Bodyguard of Lies.* New York: Harper & Row, 1975.

Johnson, Brian. *The Secret War.* London: British Broadcasting Corp., 1978.

Jones, Reginald Victor. *Most Secret War.* London: Hamish Hamilton, 1978; American Ed.: *The Wizard War: British Scientific Intelligence, 1939–1945.* New York: Coward, McCann & Geoghegan, 1978.

Kahn, David. *The Code Breakers.* New York: MacMillan, 1967. (In 1978, Kahn wrote another book, *The German Spies.*)

Lewin, Ronald. *Ultra Goes to War.* London: Hutchinson, 1978; New York: McGraw-Hill, 1979.

Randell, Brian. "The Colossus." Univ. of Newcastle Computing Lab Rep. No. 90, 1976; also in N.Metropolis, J. Howlett, Gian-Carlo Rota, Eds., *A History of Computing in the Twentieth Century* (Proc. of the Internat. Conf. on the History of Computing, Los Alamos, New Mexico, 1978). New York: Academic Press (in press).

Rohwer, Jürgen. *The Critical Convoy Battles of March 1943: The Battle for HX.229/SC122* (English translation by Derek Masters except Appendix X translated by Lt. Col. A.J. Barker), Annapolis: Naval Institute Press, 1977.

Stevenson, William. *A Man Called Intrepid.* New York: Harcourt Brace Jovanovich, 1976.

Winterbotham, Frederick W. *The Ultra Secret.* New York: Harper & Row, 1974.

The History of the JOHNNIAC

F. J. GRUENBERGER

This reprint of an early RAND Memorandum by the author describes the thirteen-year life of the JOHNNIAC computer, a Princeton-class machine designed and built at The RAND Corporation in 1953. The history presented here is based on documents and recollections of the individuals involved in the creation of JOHNNIAC.

Key words and phrases: JOHNNIAC, JOSS, John Williams, John von Neumann, RAND Corporation, Princeton-type machine

CR categories: 1.2, 6.0, 6.2, 6.3

Prologue 1979

Any piece of history must be viewed in proper perspective. JOHNNIAC became operational in 1953. This paper appeared 15 years later, and we are now viewing *it* after another decade of conditioning. The computing industry is now (1979) in production with integrated circuitry having a density of a quarter of a million active elements (e.g., transistors) per square inch, so that all the electronics of a machine like JOHNNIAC can be fabricated on a wisp of a chip—to sell for, say, $20—which will function faster and better than JOHNNIAC ever did. We thus tend to become jaded and blase when reading about the troubles of the early days, and we find ourselves wondering how they could have been so simple-minded about things that are now quite clear. No doubt it will be possible to make the same statement again in 1990. It seems that every single thing that we know today about computing had to be learned—and relearned—the same painful and expensive way. It may just be that simple re-reading of history once or twice a year can provide us with some insight, and it is insight that we desperately need.

So we are talking about a time that is over a quarter of a century ago, and you are invited to try to put yourself into the context of that time. A significant portion of the people active in computing today were not yet born.—F.J.G.

Preface 1968

The purpose of this Memorandum is to capture some history and flavor of an era. The era is that of the pioneering days of the world of computing, specifically, the thirteen-year life span of the JOHNNIAC. JOHNNIAC,[1] one of the last of the so-called Princeton-class machines, became operational early in 1953 and was retired to the Los Angeles County Museum early in 1966, thus spanning the first decade of the computer industry.

The Princeton—or Institute for Advanced Study—computers include, besides the IAS machine itself, ORDVAC, ILLIAC, AVIDAC, ORACLE, MANIAC, WEIZAC, BESK, DASK, CSIRAC, and JOHNNIAC, all built to the logic described by Burks, Goldstine, and von Neumann in the historic paper "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument."[2]

Each of the Princeton-type machines had the same general logic, but they were hand-built, individually adapted, and each had, therefore, its own peculiarities. JOHNNIAC was no exception. Its designers began with the hope of stretching the mean free time between failures and of increasing the overall reliability by a factor of ten over previous machines.

The history presented here has little direct scientific value. Technology has long since made obsolete those early machines. What value this Memorandum has rests in its bringing together

 Originally appeared as RAND Memorandum RM-5654-PR, October 1968, with the credit statement: "This research is supported by the United States Air Force under Project RAND—Contract No. F44620-67-C-0045—monitored by the Directorate of Operation Requirements and Development Plans, Deputy Chief of Staff, Research and Development, Hq. USAF. Views or conclusions contained in this study should not be interpreted as representing the official opinion or policy of the United States Air Force."

Author's address: Department of Computer Science, California State University, Northridge, CA 91330.

[1]The machine is named for John von Neumann, who protested the choice of name. John Williams settled the matter, sending him the following note:

"Ed Paxson has relayed your blushing disavowals regarding the appropriateness of the term 'JOHNNIAC.' In this matter your view just represents dispersion. If it helps any, recall that there are lots of Johns in the world."

[2]An abridged version was published in *Datamation* (September and October 1962), and an edited version appears in A.H. Taub, editor, *Collected Works of John von Neumann, Vol. 5*, pp. 34-79 (NY: Macmillan, 1963). The original text shown as Appendix B [with the orignal of this Memorandum and not reprinted here] is transcribed from the report submitted to the U.S. Army Ordnance Department [by the Institute for Advanced Study, Princeton, N.J., June 28, 1946, and appears in C.G. Bell and A. Newell, *Computer Structures: Reading and Examples* (NY: McGraw-Hill, 1971)].

some of the documents that attended the conception and construction of JOHNNIAC and some of the reflections of individuals who participated.

A Memorandum [Appendix A] of John Williams, then head of RAND's Mathematics Division, to RAND's Vice President J.R. Goldstein, without which JOHNNIAC might have met an early death, puts JOHNNIAC in perspective with other computers and pioneering efforts in the early 1950s. Shorter memorabilia and statements given in interviews are included in the text.

Funds for the development, construction, and maintenance of the JOHNNIAC were supplied entirely by the United States Air Force through its Project RAND contract. Most of the research areas explored on JOHNNIAC were supported by Air Force contract and Air Force encouragement. The Air Force is therefore an essential part of this history of JOHNNIAC.

History of the JOHNNIAC

The history of JOHNNIAC begins with the decision in late 1950 to build a computer at RAND. At that time, RAND's computing facility, in some sense the world's largest installation for scientific computing, operated six IBM 604 calculators around the clock. In 1949, when IBM's Card-Programmed Calculator emerged, RAND ordered two of them, though rumor had it that there would be only six built.[1]

In 1950, as RAND felt the need for more computing power, few electronic computing devices were operating. The Princeton machine (the first to be started according to the logical design laid out in the Burks-Goldstine-von Neumann paper) was nearly built, but not yet operating. UNIVAC I was committed but no deliveries had been made. As shown in Appendix A, three copies of the Princeton machine (at Los Alamos, the University of Illinois, and the Argonne Laboratory) were under construction.

A decision had to be reached: should RAND build or buy? And if the decision was to buy, then buy what? (There was, of course, a third possibility; namely, to continue and extend the 604-type of computing.) John von Neumann was a RAND consultant at the time, and properly pointed out that the way to go was via stored programming. Though no one—including von Neumann—could foresee the wide applicability and tremendous power of the true computer, all concerned felt that the wired-program route was a dead end.

John Williams, George Brown, and William Gunning set out on a tour to investigate what might be bought or built. The team visited the IBM plant at Poughkeepsie; the University of Illinois installation; the Moore School of Electrical Engineering at the University of Pennsylvania, where the EDVAC was under construction; and the Eckert-Mauchly Co., where BINAC was built and UNIVAC I was about to go into production.

What the team found was discouraging. A large part of their concern was with questions of reliability, safety margins, mean free time between failures (MFTBF), and operating speeds. All efforts were being bent toward devices that would have thousands of components, and these components would all have to work together. For many of the groups working on such problems, the techniques being explored were modifications of radar technology, which was largely analog in nature. To quote Gunning, "They were doing all kinds of tweaky things to circuits to make things work. It was all too whimsical."

This comment pertains to the groups that were at least forging ahead. The picture at IBM was worse. They seemed content with the 604 (wired-program) philosophy, perhaps extended with a magnetic drum. At that, the circuitry of the 604 was not reliable enough, as the RAND team well knew from having lived with six of them. In any event, the word from IBM was negative as far as computers were concerned. The word from IBM was that they had no intention of venturing into stored program machines.

The bright spot found in the tour around the country was at Princeton, where Julian Bigelow was the chief engineer in charge of building the IAS machine. Bigelow was conscious of the weaknesses in using modified radar techniques, and already had experience with many techniques that should not be repeated; for example, he was able to point out the weaknesses in Williams tube storage. The antipathy to this storage device was heightened later when Gunning spent three days a week at UCLA working on SWAC [(National Bureau of) Standards Western Automatic Computer], which used the Williams tube store (and did to its dying day).

The decision was to build, and Gunning became the project engineer of a not very formal group.

[1]Though original plans called, it is said, for only six CPCs, a formal announcement by IBM in November 1948 stated that 25 machines would be installed by the mid-1950s. Ultimately some 700 machines were installed.

RAND's decision probably had some impact on IBM's decision to swing from punched-card equipment to computers. In 1962, when Thomas Watson Jr. gave a dinner in New York for those who had figured in the IBM decision, he included George Brown, who was head of RAND's Numerical Analysis Department at the time of the JOHNNIAC decision. Mr. Watson's theme was "We were pushed into it, and these [guests] were the people who pushed us." His other guests were: J. Presper Eckert and John W. Mauchly, co-inventors of ENIAC, and (in the period 1948–1950) the designers of UNIVAC I; the official of the Metropolitan Life Insurance Company who ordered a UNIVAC I; the IBMer who had taken the stand, against the current company policy, that IBM should swing to computers. Each guest received an expensive watch as a token of IBM's gratitude.

The quality of JOHNNIAC's design must be measured against the context of its time. It is tempting to disparage the early models of every technological innovation—the Model T Ford can appear crude in comparison to a modern car. In terms of what could have been built (using the tools available at the time of construction) and what could have been properly used if it had been built, JOHNNIAC measures up extremely well. It had a useful life that just covered the first decade of mass production, during which time some 20,000 other machines were built.

At the time of JOHNNIAC's design, modern techniques (e.g., magnetic storage devices) were not available, nor were modern ideas of machine organization (e.g. indexing, or indirect addressing), so the design as seen today appears somewhat primitive. It is important, however, to bear in mind the context of computing needs of the day. IBM, for example, was about to propose that 18 of the 701 computers might saturate the country's computing needs, as indeed they would have if they had all been available in 1951. (It would be a fair estimate that one Control Data 6600 can process in a few months all the computation that *all* the 18 701's performed in their lives.)

The point is that JOHNNIAC represented the state-of-the-art of its day. It could have been made better, perhaps, but could its users have capitalized on the improvements? In the period of peak use of the machine (1954–1957), computers were invariably used by professional computer people, and the philosophy of the day called for a user to have the entire machine. Today's scheme of operation for large machines revolves around monitor routines; the user may have nearly all the machine, but the master control is performed by the monitor. In 1955, people were just beginning to explore the idea that a computer could be efficiently shared by many users.

The term "state-of-the-art" carries a different connotation today from what it did during the fabrication of JOHNNIAC. Today it implies "If anyone had done it, we can do it, too, and perhaps something more." In 1951, state-of-the-art meant something a little different, in that one might be aware of certain developments and yet find it difficult to adapt them to a particular use.

Consider, for example, the decision to put punched-card input/output gear on JOHNNIAC. There were then only two American firms making equipment to handle cards: IBM and Remington Rand. The Remington machines were undesirable for many reasons, an important one being that RAND was loaded with standard IBM punched-card equipment, and the decision to use cards revolved around the availability of this auxiliary equipment. But, unfortunately, IBM did not list in their catalog items described specifically as "card reader into computer" or "computer to card punch."

The 077 collator and the 523 summary punch were candidates, respectively, to be a card reader and a punched-card output device. To perform the proposed marriage between these standard devices and the computer, some changes had to be made to them. In 1951, this simply was not being done. Even the installation of one toggle switch on the 523 caused a heated conflict between RAND's engineers and IBM which ran for many months. Eventually it was resolved with grudging permission for the modification.

Another aspect affecting the decision to use punched-card equipment for input and output was the feeling of many people that punched cards were about to be phased out, that the swing

Fred Gruenberger, while at The RAND Corporation, was associated closely with various studies made with the aid of JOHNNIAC, and at the time of writing the Memorandum was a RAND consultant. He is now a Professor of Computer Science at California State University, Northridge, and the editor of the magazine Popular Computing.

was to photoelectric readers and punched paper tape. There was also a common misconception that, in scientific computing, input and output were relatively unimportant, since most problems would be compute-bound anyway. The people on the JOHNNIAC project felt that a punched-card reader was relatively reliable, was time-tested, and was the fastest device for the cost. Their judgment in this matter may have had an effect on the entire field.

John von Neumann

The design goal for JOHNNIAC was an improved copy of the Princeton machine. The MFTBF was to be significantly longer than 10 minutes (a time that was considered to represent the state-of-the-art, although the other Princeton machines were already bettering that figure), and the overall reliability was to be increased by a factor of ten over all previous designs.

Today it is difficult to appreciate a MFTBF of 10 minutes. Equipment of as far back (now) as 1961 is so reliable that some people argue that machines used for training should have a built-in failure switch, so a student can experience, on demand, what a machine failure is like. MFTBFs of six months (in the electronic portions of computers) are not uncommon, leading a student to believe that a machine can never fail.

On the other hand, one is tempted to ask how any computing at all can be accomplished on a large problem with a MFTBF of 500 seconds or so. The answer is that the users of that day had to be extremely clever. They were faced with the availability of what seemed to be fantastic computing power in short bursts. The game reduced to outwitting the machine failures that were due and payable without warning. The people who became adept at that game are the senior citizens of the computing world today. The thought of having a machine that could be depended on to give up to an hour (now and then) of trouble-free computing was stimulating.

For JOHNNIAC, one important change from earlier designs was a closed-cycle air-cooling system surrounding the electronic circuits. The idea was to enclose the chassis completely and pump cold air through the enclosure. This was quite feasible. Unfortunately, during the early days of operation, the glass doors were open a good deal of the time for trouble shooting. For a while, the machine was known as "Pneumoniac."

The use of more-or-less standard punched-card devices for input and output was a departure from the designs of previous machines. Another innovation was the use of octal notation, a "shorthand" for pure binary, as a convenience to the programmers. Earlier machines had made some use of hexadecimal notation—what today might be called 4-bit bytes.[4]

JOHNNIAC was designed to be maintained. Among other things, this meant that all tubes were readily accessible; that tube-heater voltages could be monitored remotely; that heater-cathode leakage could be monitored with the machine in operation; and that voltage levels could be varied at the console for subsets of the machine.

In addition, the circuits in JOHNNIAC were to be interlocked in such a way that the precise shape of signals was not important, in the following sense: each step furnished a "done" signal to allow the subsequent step to proceed. Thus if for any reason the machine hung up, in theory the bad condition was right there. In previous machines, diagnosis was possible under dynamic conditions only with an oscilloscope. In principle, the JOHNNIAC would be able to run at any speed from DC to its maximum. This idea was felt to be a big step forward. When the Selectron store[5] of the machine became operational, checking facilities were added so that the computer itself could be used to exercise and check all of the storage.

Construction of JOHNNIAC began with Gunning in charge of the hardware—"strictly a nuts and bolts sort of thing," as he puts it. Ideas were

[4]Ideas never seem to die in the computing world. After many years of "Thank heaven, at least we don't have to deal with hexadecimal notation any more," it's back with us in IBM's System/360. Similarly, many people were pleased at the passing of JOHNNIAC's two-instructions-per-word format; but it keeps coming back, e.g., in the Russian BESM-6.

[5]See Appendix A for a discussion of the adventures with this novel storage device.

accepted from any source. A key one was Julian Bigelow's: to eliminate capacitance coupling between circuits. The Princeton machine was nearly completed by that time—"It looked like a 40-cylinder diesel engine."

The following appeared in a RAND computer sciences department newsletter in 1952:

> Discussions are in progress with regard to the console. Several display schemes and methods for entering numbers into the machine are being considered. Probably there will be an operator's console presenting to him only as much as he needs to play the machine, and a maintenance console which reveals the deepest secrets of the whole JOHNNIAC. No other machine can make this statement: Our console is human engineered.
>
> JOHNNIAC will definitely be the most completely protected machine ever devised. The present plans for supervisory control will take care of the machine in event of voltage failure, refrigeration failure, fuse burnout, and all else. In addition to shutting down the machine, an alarm will be sounded and a tell-tale light will tell who "do-ed" it. The precise nature of this alarm is not yet settled; many diabolical devices, all directed toward the best interests of the operator, are being considered.

Other safeguards were planned. Wes Melahn remembers:

> Bill Gunning asked Gardner Johnson, one of the staff engineers, to procure a switch for the main power supply for JOHNNIAC that would be safe from accidental movements. Gardner complied and made his selection with great conservatism and regard for mechanical as well as electrical characteristics. The switch had a lock and key that would have discouraged any tampering and no doubt made the Yale people happy. No accidental movement was likely either, because both hands of a strong man were clearly required to turn the switch. All who saw the massive construction could not doubt its mechanical strength, and the big copper elements of the connectors certainly looked reliable.
>
> I don't believe Gardner's switch was used for JOHNNIAC. As I recall, Gunning quietly put the

Keith Uncapher at the JOHNNIAC Console

switch back on the market, and it is now probably used to connect some major city's electrical system to its generators.

In 1952, as hardware for JOHNNIAC neared completion, the prospective users faced some of the basic philosophic questions of computer design that von Neumann had considered in his paper.

For example, von Neumann discussed the pros and cons of having floating-point hardware (as opposed to subroutines for floating point) and concluded that the cost of the hardware could not be justified. Again, he dwelt at length on the philosophy of considering the machine as inherently integral (i.e., all numbers are integers) or fractional (i.e., all numbers have their binary point at the far left).

JOHNNIAC users in 1952 faced other problems as well. Would it pay to include special operation codes to facilitate subroutine linkage? What form should the divide operation take?—indeed, is a separate divide operation essential at all? To what extent should automatic rounding be built into the multiply commands?

Meanwhile construction progressed. Something of the atmosphere of that phase of JOHNNIAC's history emerges from Cecil Hastings' memorandum, "JOHNNIAC Progress Report," dated August 8, 1952.

> As is fairly evident to anyone who goes by the zoo,[6] the main frame for the JOHNNIAC is ready to receive registers. Bob Rumsey, who has been working with Mike Stobin to wire the filament transformers which supply power to heat vacuum tubes, has formed a private operation outside where he is holding down floor space vacated by IBM files. We promise to have this auxiliary activity (you might call it Rumsey's Rump Session) replaced by bona fide JOHNNIAC ventilation.
>
> Gan Baker has been given the awesome responsibility of Chief Inspector. What this means in essence—we know where to point the finger—anything that goes wrong is, of course, Gan's fault. Under Gan's direction the shop has produced all of the chasses of the adder, the digit resolver, the accumulator and the MQ. Two memory registers are completed; two more will be completed in two weeks. Two clear and gate drivers have been completed.
>
> What all this adds up to is, that if Mike Stobin and Willis Ware who have been dealing with the ventilation engineers can come through with the ventilating equipment in time, it is very likely that we can have a smoke test of the arithmetic unit (less control of memory) in the JOHNNIAC main frame in October. The goal of the test will be to connect the A and MQ for end-around shifting (7.5 order) and let the machine shift a set of digits all day while we hammer on the frame and wiggle wires. Applications for wire wigglers are now open.

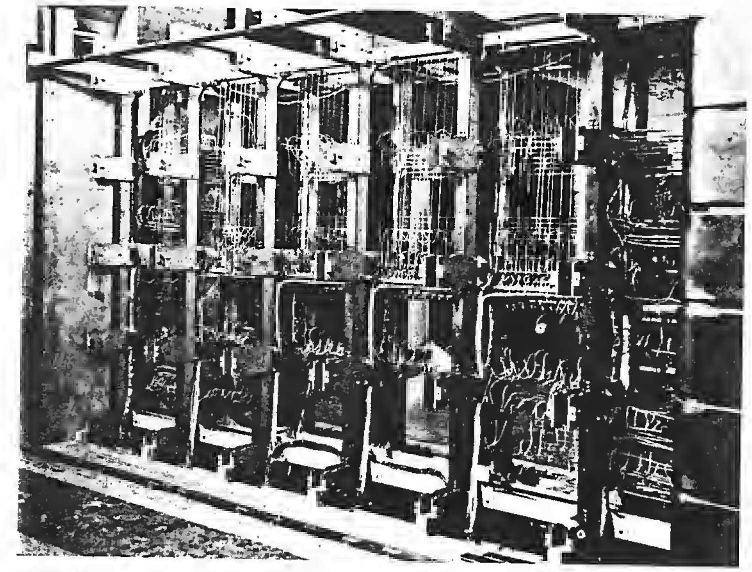
Construction of JOHNNIAC continued during the first quarter of 1953. During this period, The RAND Corporation moved from its early quarters to its present site. Shortly after the move, JOHNNIAC Junior's[7] arithmetic unit was placed in operation for the first time. "During the time it was tested, something over a billion operations (adding, subtracting, and shifting) were carried out without a single error." (How time flies. The IBM 1130—a desk-sized computer costing around $30,000—can perform a billion such operations in a little over two hours—and is *expected* to have no errors.)

Seminars were held regularly, and sample problems were coded and analyzed. The tradeoffs implied in the questions above (e.g. built-in floating point) were debated at length, and those who were writing codes were beginning to see the rich possibilities in being clever and ingenious in their use of instructions. All of this took place in an atmosphere of uncertainty, since no one had a good feel for what problems would eventually be tackled. Background and experience were conditioned by years of what today would be called straight-line programming on the wired- and card-programmed machines. The beauty of looping with address modification and subroutines was still novel. What's more, the computer was still looked at as a pure calculating device; the thought of using it as a symbol-manipulating device was yet to be explored.

In 1968, it is rare for the designers and the users of a new computer to meet. Few, if any, modern-day programmers have had the experience of discussing with design engineers the kinds of instructions the machine should have. In 1952, this was a give-and-take process of programmers saying "Can you do this?" and engineers replying "Yes, but this is easier." The process would continue until a compromise was reached on each class of instruction. Don Mad-

[6]"Zoo" refers to a caged area built inside the cleared facility.

[7]A preliminary, quarter-sized computer, built as a prototype.



JOHNNIAC core memory partially installed (produced by Telemeter Magnetics, Inc. under a subcontract)

den recalls that many hours were spent converging on the characteristics of JOHNNIAC's shift commands.

JOHNNIAC went on the air in the first half of 1953,[8] using the Selectron store. Shortly thereafter, a contract was let with International Telemeter (later Telemeter Magnetics) for the construction of a core storage. The specifications for this store (the first commercial 4096-word core store) were written by Gunning and Willis Ware.

Over the years, the hardware of JOHNNIAC was improved and updated. The original Selectron store, for example, was replaced by a magnetic core store in early 1955. The vacuum tube circuitry was somewhat replaced by transistors, and so on. A memorandum of January 1956 indicated that despite mechanical problems associated with the drum and the punch-card gear, JOHNNIAC was one of the most reliable computers then in existence, consistently ahead of RAND's 701 in this respect. For the electronic components, the MFTBF was never under 100 hours. It has been said that despite the word "electronic" in electronic computers, such systems still have many mechanical components which are orders of magnitude less reliable than electronic ones. That statement was true in 1955 and is still true today, although the reliability levels of all components have shifted significantly upward.

From 1957 on, those in charge of the machine were constantly facing the choice of further extending the machine's capability or of abandoning it. Should central storage be expanded beyond the original 4096 words? Should magnetic tape capability be added? (A 12,288-word drum was added to the machine shortly after the installation of the core storage.) How about indexing? Or built-in floating-point operations? These features were not added, in part because they were available on RAND's commercial (IBM) computers. One feature, indirect addressing, was added in 1963. In other respects, the logical design of the machine remained much the same during its life. An on-line plotter was added to the machine in 1958, an ANelex line printer was

[8]Curiously, in all the mass of printed matter concerning JOHNNIAC, there is no clue as to the day the machine became operational.

LIST OF JOHNNIAC OPERATIONS – MARCH 16, 1955

Operation Codes are Base Eight Numbers. Notes are Base Ten

000		Proceed to next order in sequence	004	LM	Clear MQ, M → MQ
001	TNL	If A < 0, c → left command in M	005	TNR	If A < 0, c → right command in M
002	TPL	If A ≥ 0, c → left command in M	006	TPR	If A ≥ 0, c → right command in M
003	TFL	If overflow, c → left command in M	007	TFR	If overflow, c → right command in M
010	TRL	c → left command in M	014	TRR	c → right command in M
011	T1L	If T_1 on, c → left command in M	015	T1R	If T_1 on, c → right command in M
012	T2L	If T_2 on, c → left command in M	016	T2R	If T_2 on, c → right command in M
013	T3L	If T_3 on, c → left command in M	017	T3R	If T_3 on, c → right command in M
020	RA	Clear A, M → A	024	A	M + A → A
021	RS	Clear A, – M → A	025	S	– M + A → A
022	RAV	Clear A, | M | → A	026	AV	| M | + A → A
023	RSV	Clear A, – | M | → A	027	SV	– | M | + A → A
030	MR	Clear A, M · MQ rounded → A	034	MB	M · MQ + 2^{-39} $[A + \frac{1}{2}(1 - {}_1A_1)]$ → A and MQ
031	MNR	Clear A, –M · MQ rounded → A	035	MNB	–M · MQ + 2^{-39} $[A + \frac{1}{2}(1 - {}_1A_1)]$ → A and MQ
032	M	Clear A, M · MQ → A and MQ	036	MA	M · MQ + 2^{-39} A → A and MQ
033	MN	Clear A, –M · MQ → A and MQ	037	MNA	–M · MQ + 2^{-39} A → A and MQ
040	DS	A ÷ M → MQ, r → A	044	D	(A + 2^{-39} MQ) ÷ M → MQ, r → A
041	DNS	A ÷(– M) → MQ, r → A	045	DN	(A + 2^{-39} MQ) ÷(– M) → MQ, r → A
050	ST	A → M	054	SAB	${}_7A_{18}$ and ${}_{27}A_{38}$ → ${}_7M_{18}$ and ${}_{27}M_{38}$
051	SOL	${}_0A_6$ → ${}_0M_6$	055	SOR	${}_{20}A_{26}$ → ${}_{20}M_{26}$
052	SAL	${}_7A_{18}$ → ${}_7M_{18}$	056	SAR	${}_{27}A_{38}$ → ${}_{27}M_{38}$
053	SHL	${}_0A_{19}$ → ${}_0M_{19}$	057	SHR	${}_{20}A_{39}$ → ${}_{20}M_{39}$
060	STQ	Clear A, MQ → A and M	064	AQS	MQ + A → A and M
061	SNQ	Clear A, – MQ → A and M	065	SQS	– MQ + A → A and M
062	SVQ	Clear A, | MQ | → A and M	066	AVS	| MQ | + A → A and M
063	SNV	Clear A, – | MQ | → A and M	067	SVS	– | MQ | + A → A and M
070	SRC	Clear MQ, shift A right n places Zeros into A_0.	074	SRH	Shift A right n places. Zeros into A_0.
071	CLC	Clear MQ, circular shift of A and MQ left n places. Couple MQ_0 to A_{39}, A_0 to MQ_{39}.	075	CLH	Circular shift of A and MQ left n places. Couple MQ_0 to A_{39}; A_0 to MQ_{39}.
072	LRC	Clear MQ, power shift A and MQ right n places. Couple A_{39} to MQ_1. ${}_0A_0$ to → ${}_0MQ_0$.	076	LRH	Power shift A and MQ right n places. Couple A_{39} to MQ_1. ${}_0A_0$ → ${}_0MQ_0$.
073	LLC	Clear MQ, power shift A and MQ left n places. Couple zeros into MQ_{39}; MQ_1 to A_{39}.	077	LLH	Power shift A and MQ left n places. Couple zeros into MQ_{39}, MQ_1 to A_{39}.
100	SEL	Select input output Address part of 100: XXX0 Pri. Feed Reader; XXX1 Sec. Feed Reader; XXX2 Feed Punch; XXX3 Feed Punch and echo; XXX4 Select Printer and space; XXX5 Select Printer	104	DIS	Display
			105	HUT	Hoot
			106	EJ	Eject page
101	C	Copy			
110	RD	Read drum words to M and memory addresses following numerically. Denoting MQ as xxx $f_1f_2f_3f_4$ dpb $l_1l_2l_3l_4$, the f's determine the first drum address and the l's the last drum address. d selects the drum; p, the position of the heads; and b, the band to be read.			
111	WD	Read M and words in memory addresses following numerically to drum. MQ has the same significance as in 110.			
120	ZTA	Clear A to Zero	124	PI	M I A → A
121		Clear A	125	NI	$\bar{}$ M I A → A (– denotes digit inversion of M).
122		Clear A	126	PMI	| M | I A → A
123		Clear A	127	NMI	– | M | I A → A
130	HTL	Halt c → left command in M	134	HTR	Halt c → right command in M
131	H1L	Halt if H_1 on; c → left command in M	135	H1R	Halt if H_1 on; c → right command in M
132	H2L	Halt if H_2 on; c → left command in M	136	H2R	Halt if H_2 on; c → right command in M
133	H3L	Halt if H_3 on; c → left command in M	137	H3R	Halt if H_3 on; c → right command in M
140					
150					
160					
170					

DEFINITIONS

A	Accumulator	${}_{20}A_{39}$	Digits in position 2^{-20} through 2^{-39} of the word in A.
MQ	Multiplier Quotient Register	c →	Control goes to
M	Word in the M^{th} address in Internal Storage	I	Logical (digit by digit) product or Intersection.

added in 1955, and an improved ANelex printer in 1958.

A major programming effort on JOHNNIAC was the Linear Programming Code, written by Bill Orchard-Hays and Leola Cutler, which was active until 1961. (There is an interesting sidelight. During its life, the Linear Programming Code was persistently plagued by an obscure drum error that no one could diagnose. With the advent of JOSS (the JOHNNIAC Open Shop System)[9] years later, Dick Stahl decided to double the capacity of the drum by doubling the recording density. Dick asked Mort Bernstein to write a special test routine for the drum, and this routine uncovered the logical bug in the original drum circuits.)

During most of its active life, JOHNNIAC had a central core store of 4096 40-bit words, supplemented by the 12,288-word drum. There was a repertoire of 83 operation codes including eight halts and a hoot. The single-address logic called for 7 bits in an instruction for the operation code and, of course, 12 bits for the address. A complete instruction, then, could be contained in 19 bits, and it was logical to pack two instructions into one 40-bit word.

The illustration shows the list of JOHNNIAC operations as it appeared in 1955, after many iterations. It should be noted that the sheet defines each operation code precisely, offering the user a condensed machine manual on one page (another innovation with JOHNNIAC).

Input to the machine was provided through:

(1) A keyboard (closely resembling the keyboard of the then-current Friden calculator out of which it was constructed) plus some operating buttons.

(2) A punched-card reader, made by modifying an IBM 077 collator.

Output was provided through:

(1) A card punch; an IBM 523 summary punch, suitably modified.

(2) A line printer. After 1959 this was an ANelex printer capable of up to 1200 lines of numeric printing per minute, which was fast even for 1965. At that, the printing speed could have been set at 1800 lpm, but at reduced quality.

(3) A loudspeaker, connected to the machine's operation decoder.

(4) Console lights, including a set of Lucite edge-lighted figures that gave the contents of certain registers in ordinary Arabic (octal) numerals.

These peripheral devices—particularly the punched-card gear—were not designed for JOHNNIAC. Consequently, a body of programming and operating lore built up around the machine, with local house rules like "never hit LOAD with the IR selected," and "don't hit STOP while the card reader is reading unless you want to lose a card." Most of these rules related to timing considerations. The troubles were recognized, but no one could or would remove their causes.

The loudspeaker was a simple and inexpensive device to make the action of the machine audible, and was intended originally as a means of communication with the computer's operator. However, it was noticed that by proper choice of computer instructions (meaningless computationally), the sounds could be made meaningful in the form of music.[10] A lot of extra-curricular programming effort went into the construction of card decks that would cause the computer to "play" things like "Jingle Bells" or "Hail to the Chief" (in case the President dropped in). Eventually, of course, this process was mechanized by the writing of a master program[11] to make it easy to transcribe any given piece of music. More important, the condition of things (e.g. a tight loop) could be sampled dynamically, and reached the user's mind much faster than lights, bells, or printed messages. All in all, this was a neat and clever idea, and still is: nearly every college computing center rediscovers it independently. (With today's faster machines, an FM radio near the console does the job without having to tap into the computer's circuitry.)

Since JOHNNIAC dates back to early explorations of stored programming computing, much of its first programming was done in absolute octal. Very soon, however—around 1954—a symbolic, relative assembler was written by Jules Schwartz (of JOVIAL fame). Associated with this assembler was a system of relative binary library routines that formed the predecessor of what is now called relocatable code. In 1955 Cliff Shaw produced a load-and-go assembler (EASY-FOX) that added

[9]JOSS is the trademark and service mark of The RAND Corp. for its computer program and services using that program.

[10]This had been noticed earlier on the SEAC [(National Bureau of) Standard's Eastern Automatic Computer].

[11]A true compiler, written very early in the days of compiler writing.

the feature of local symbols. EASY-FOX had another feature that was somewhat novel for its day; it was written in its own language.

In the days of JOHNNIAC, conversion between base 10 and base 8 was a chore that programmers performed daily. Wall charts of conversion tables were a help (every programmer's office was papered with them), but an octal desk calculator was nice to have, particularly if it had 13 banks so that it would simulate the JOHNNIAC word format exactly. A government agency was found to have declared two 13-bank Monroes surplus, and RAND successfully bid on them ($79 and $104). The cost of converting one of them to base 8 ran around $500. The machines are probably still in use.

A package of floating-point subroutines was developed; and in 1958 an interpretive coding system, QUAD, was added to the library, followed by SMAC, a small compiler. Both QUAD and SMAC, though somewhat trivial by today's standards, were notable for being foolproof. In both coding systems, open-shop users found little need for mothering; i.e. all troubles were reported to them by explicit printed error messages or, in the case of an endless loop, by an operator's message. The manuals for these systems were short but complete. The systems were active for about two years, until open-shop work was converted to FORTRAN II.

At the decommissioning ceremonies J. Cliff Shaw at the console; Irwin Grenwald, far left; Edward Bryan, center.

The decision to build JOHNNIAC was made at a time when no commercially mass-produced machine was available. Coincidental with JOHNNIAC's going on the air in 1953, the IBM 701 appeared. From then on, excellent machines were always available, and RAND maintained a large computing facility with commercial machines: the 701 from 1953 to 1956; the 704 from 1956 to 1959; and various other machines (7090, 1401, 305, 7040, 7044 and System/360s) subsequently.

Like all the early computers, JOHNNIAC was associated with legendary anecdotes, such as the story that it was afraid of the dark. As Mort Bernstein tells it:

> It must have happened around 1959. Harriet Pierson was running part of payroll at night on JOHNNIAC and would have great difficulties with the punch—an unreasonable high rate of echo-check failures. When the job was rerun during the following day, no such trouble was encountered. After a bit of digging it was determined that the only difference in the operation was that the JOHNNIAC machine-room lights were turned out at night when the run was made and no one was in the room. The only conclusion that could be drawn was that JOHNNIAC was afraid of the dark, and indeed it was true. Neons had been used as active elements in the I/O lines and some had aged and become de-ionized so that they would not conduct without the aid of some external radiation—namely, light. So it came to pass that JOHNNIAC was provided with its own lights independent of the room lights and never had to be in the dark again.

From 1957 on, JOHNNIAC was considered essentially a "free" machine, easily maintained at low cost and quite useful for computer experimentation. On February 18, 1966, at the decommissioning ceremonies for JOHNNIAC, Willis Ware recalled some landmark uses:

> In the earliest days of 1954, most programming was done in machine language and in absolute octal at that. In 1955 Jules Schwartz wrote the first assembly routine for JOHNNIAC, and Cliff Shaw produced a revised assembler in 1956. Then came QUAD, an interpretive

> programming system, and SMAC, a small compiler. Each was noted for being foolproof. The non-professional programmer could use these systems comfortably; his errors would be reported to him in great detail by the machine. There were other significant contributions to the programming art as well; among them were items with such names as EASY-FOX, CLEM, JBL-4, J-100, MORTRAN done by Mort Bernstein, and Load-and-Go.
>
> In the late fifties, the nature of JOHNNIAC's task changed. The rental equipment from IBM carried most of the computing load from the RAND staff. JOHNNIAC became a free good; its time was available for research use. The cost of operation was sufficiently low that one need not be concerned about using large amounts of machine time. Much of its time was consumed by research on the general questions of artificial intelligence and the initials NSS came to be closely associated with JOHNNIAC. These are the initials of Allen Newell, Cliff Shaw, and Herb Simon who used the machine extensively for research. During this period came such achievements as:
>
> •List structures, list processing techniques and their embodiment in such languages as IPL-2, -3, -4;
>
> •Chess playing routines such as CP-1 AND -2;
>
> •Theorem proving routines such as LT—the Logic Theorist;
>
> •The general problem solver—GPS;
>
> •The assembly line balancer of Fred Tonge.
>
> Most recently JOHNNIAC has been the research tool which made possible both of RAND's current highspots in computer research. The initial experiments on graphical input-output terminals was done on JOHNNIAC and from that has come the successful development of the RAND Tablet. Finally, JOHNNIAC has made JOSS possible. JOSS is the JOHNNIAC Open Shop System which provides each of its time-shared users with a typewriter connection from his office to the machine. Those who know JOSS and perceive the friendliness of its help and reaction feel strongly that systems such as it will be one of the prominent, if not exclusive, ways of computing for the future.
>
> Certainly, it is fitting that a machine with the stature of the JOHNNIAC should have completed its career as a research vehicle, dedicated to improving and extending the technology and art which it helped inaugurate.[12]

[12]From "JOHNNIAC Eulogy" by Willis H. Ware (RAND Corp. document P-3313, March 1966, pp. 11-12).

Willis Ware giving the signal to "disable"; W. Gunning, left.

In a press release dated February 18, 1966 (written by Shirley Marks), RAND announced the demise of JOHNNIAC:

JOHNNIAC 1953-1966

Friday, February 11, 1966, as it must to all men—and machines—the end came to JOHNNIAC, member of a distinguished family of computers known as Princeton-type machines. This noble line of electronic brains was sired by the human brain of mathematician John von Neumann, for whom JOHNNIAC was affectionately named.

The end came to JOHNNIAC in the same room at The RAND Corporation in which, more than twelve years earlier, its neons first flickered into life. JOHNNIAC had entered a world which saw the computer only as a mechanical extension of man's hand on the keyboard of a desk calculator. With the brashness of youth, with the knowledge of its uniqueness, with the spirit of a pioneer, JOHNNIAC has been credited with leading the way to the modern concept of the computer as an information processor—an electronic extension of man's mind, helping him to design, to plan, to judge, to decide, to learn.

As the end came, from nearby rooms was heard the busy chatter of JOHNNIAC's sophisticated descendants. Absorbed in the wonder of their mass-produced cores and graphic displays, of their systems and languages, they seemed unaware of the drama drawing

to a close, of a memory fading, a pulse unsteady. And finally, power failure: JOHNNIAC had been unplugged.

Friday, February 18, 1966, final ceremonies were held for JOHNNIAC. Many friends of early days gathered, but not to grieve. There were no flowers: only coffee, cake, and memories.

Enshrinement will be in the Los Angeles County Museum.

APPENDIX A

Memorandum Dated 12 June 1953
From John Williams (Then Head of the Mathematics Division of RAND)
To Vice President J. R. Goldstein

I was inclined to ignore the silly question about the JOHNNIAC—i.e. should we, in view of the reported success of IBM's 701 Calculator, chop it into little pieces and put it down the drain?—but I've decided to assume that you are basically sound (in fact, solid as a rock in places) and therefore capable of relearning the gospel. I will discuss the 701, and then the JOHNNIAC. Since I've never seen either type successfully add two and two and since (crossing myself devoutly) I am not an engineer, this account will perforce be free from a lot of very stuffy details—but suited to a front-office type.

The 701 looks like a typical success story in the field of big business. After sitting on its hands and patents for twenty-five years or so, IBM finally noticed that the world was beginning to act as if it could and would pass it by. I believe that RAND's decision to build its own computer may have been the final wasp sting which goaded them to their feet. It went off at a dead run, pouring in engineers by the hundred and money by the bucket, resolved to build the best possible machine in the shortest possible time—there weren't any RAND people around to explain that this is meaningless. The machine would have everything—high speed arithmetic, high speed electrostatic storage, medium speed drum storage, low speed tape storage, and organ music. The state of the art was pretty sketchy in all these fields at the time—the general principles were known, but not much practical hardware existed—so they had to lick all components into shape and, simultaneously, perform the messy marriages. If the 701 lives up to advance billing, they have done just that; in the remarkable time of two to three years.

By hammer and tongs, but they did it. For example, the only existing high speed parallel memory device was the Williams tube. This had some fundamental weaknesses which—if you were smart enough, careful enough, willing to nurse, and philosophical enough to accept morning sickness—one could live with. This had been developed into a fairly practical parallel memory device (taking off from F.C. Williams' serial memory machine) by the early birds in the Princeton-type-machine field. Aside from nursing care (expensive maintenance, son), the practical significance of the presence of this leprous element in the machine is that everyone who sits down to do a problem must be aware of it and be prepared to be just a little cagey, depending on the *problem* ; for the blips fade in a fraction of a second and if the problem requires that you re-use a number before the blip is regenerated, you get the wrong answer. It is as if a desk calculator would fail any time the 7th, 8th, and 9th places in a 15-digit number happened to be a 3-digit prime—in other words, a completely irrational thing to have to contend with: it just isn't decent for the operator to have to worry about how the machine is built. However, good design and technology, plus thoughtful use, lead to a machine which can do a lot of fine computing. Even the Princeton-types (with a laboriously selected 5% of garden variey cathode ray tubes) are pretty prodigious machines. IBM picked up this device and—knowing that the fundamental weaknesses could not be eliminated—spent (it is said) a million dollars developing an improved cathode ray tube which would work somewhat better than the garden variety. They then traded away some speed by putting in three repainting cycles for each action cycle, with the result that

the 701 programmer can forget about read-around ratio. The mean free path between errors is of the order of half an hour. For contrast, consider the horrible example, the SWAC: our programmers learned through bitter experience that, to have reasonable assurance the memory was behaving, they had to cause it to spill everything out onto cards for examination *after each minute of computing.* (The SWAC is presently edging up to a mean free path of ten minutes and to the barrier due to inferior tubes.) That it is worthwhile to compute at all under such limitations testifies to the speed of the present generation of machines compared to the last generation; the work that can be done in this minute compares favorably with an eight-hour shift on a CPC; and the mean free path between CPC errors is estimated at eight hours.

Because the 701 was developed and built in an expensive way, it is a rich man's machine. You can rent time on one for $300 an hour (you supply all personnel except maintenance crew), or you can have exclusive use of one, for a forty-hour week, for $190,000 a year (plus $25,000–$30,000 for installation). If it needs maintenance (which it for sure does), this is thrown in free, except that the maintenance is done during the same forty hours that are yours for computing; so you compute during what is left of the forty hours after repairs are made. Things are much better for second and third shift rental—if you have any money left—for these shifts together cost only $190,000, and you get a pro rata reduction for hours lost to maintenance. So the full-time rental for a 701 is $380,000 a year; of course, this is the basic machine—if you want fancy extras, like another tape or violin music, you pay extra.

For machines which have much the same mission and superficially, the same capability, the 701 and the JOHNNIAC have been conceived and built in remarkably different ways—it is sort of like the difference between a gold-plated three-dollar alarm clock and a Vaćheron chronometer in a simple titanium case. One major difference was our decision to start with a very simple machine. We plan to have every useful and convenient feature on JOHNNIAC—in the fullness of time—but our first goal has been to start with the simplest useful machine and add the features as they become solid. For instance, tapes can be beautiful, but we didn't want the headache of developing them—when they aren't quite right you may get confetti all over the building.

We naturally—being itchy—wanted our simple machine soon, but not as soon as possible, for other values came first—even an administrative one, for we decided it was best to build it out of the Numerical Analysis running budget and with permanent personnel, to the practical limit; we have thus traded some time for organizational stability, and probably for a better thought-out machine. Of our technical values, reliability came first. This is practically a fetish with us. We are dedicated to the proposition that when you ask a digital machine to estimate the sum of two and two, it should *always* (not just frequently) say "four"—unless Hoover Dam breaks, in which case the machine should turn on a red light (powered, doubtlessly, by a simple nuclear pile) and refuse to discuss arithmetic. It should do its work at a rate of an interesting number of thousands per second, and it should do it for hours on end without dropping a stitch. We also wanted it to be easy to live with—in no way tricky to operate, requiring little maintenance, and trouble being easy to identify and correct. The basic Princeton-type machine was wonderfully adaptable to our goals and, by virtue of being late-comers to the field, we were able to bypass some of the inevitable early mistakes. Moreover, since our engineers did not have to do the fundamental work on the design of certain major units, they have devoted their considerable talents to cleaning it up and putting bonded guarantees on every slightly fishy thing in the originals. We have also, I believe, been blessed with much better shop work (both electrical and mechanical) than any other Princeton-type machine. The net result is that it will probably be years before anybody produces a better machine of this type than ours. As an example of a detail which we believe will pay off: the builders of Princeton-type machines expect so little trouble that they employ about one fuse—to guard against the 440-line surging to 880, I presume (I told you I wasn't an engineer)—whereas JOHNNIAC has a panel containing hundreds of fuses; so in the event of trouble the damage will be very local and instantly identifiable. Another example: the original machines were so tightly designed as to have little flexibility for meeting afterthoughts and new developments; as a result they had to hang stuff informally on the outside and inside before they finished Mark I; whereas we (smug) allowed for twice as many tube-socket positions in standard chassis locations, and we also provided several uncommitted registers. Ours is the

only machine with a closed cycle air conditioning system. In short, we expect to have the right things in the box, to have room for them, and we expect it to work until hell freezes over—and when hell does freeze over we expect to thaw it without undue delay.

The Princeton machine was conceived about the notions of reliability and speed. The original intent was to couple the present type of arithmetic unit with a high speed Selectron memory of 1024 words. However, RCA became interested in television and never put in the development time needed to debug the tube, so the builders of Princeton-type machines were put in an unhappy position: they could choose between Williams tubes and mercury delay lines. Both were messy technologically and neither was satisfactory logically; they chose Williams tubes as the lesser evil and wore a silly grin so people would judge them to be happy. How happy they really were may be inferred from the advantages they had had to forego: In the Selectron a particular slot in the memory is selected by digital (rather than analog) means, and the output signals are a thousand times larger than those in Williams tubes and delay lines—facts intimately related to sensitivity to noise and therefore to reliability. Immediately after we decided to build JOHNNIAC, RCA showed signs of life in Selectrons; they began a small production of 256-spot tubes for the Air Force. This looked like our boat, for we could get 512 words by running a double bank of these—and we much preferred this to 1024 words on Williams tubes—which proves that (a) we are not hogs and (b) we practice what we preach on reliability. Further, since all Princeton-types were going to Williams tubes, in the interest of getting finished, it looked as though we could contribute to the field and thus pay our passage, by exploring something different; Aiken of Harvard recently complimented us on this aspect of our choice. So we abandoned plans for Williams tubes and placed a sizeable order for Selectrons in the expectation that, thus encouraged, RCA would lean into the harness, push development and production, and prepare to welcome all the frustrated computer people who had been living (reluctantly) on Williams tubes and delay lines. There was a high level confab at this juncture between RCA and IBM regarding Selectrons, which made it look as though they might hit a really big market. So RCA took its engineers off black-and-white television and put them on color television, and hired the mothers-in-law of two deserving employees (the Chairman of the Board and the President) to make Selectrons for us. Besides wanting to sell us the rejects, the price of the tubes was remarkably arranged so that the more we bought the higher the unit cost became. There is even some question that the tubes really meet RCA specifications (and they are not bound because it is a development contract).

Like my late lamented gall bladder, the above somewhat unsatisfactory situation existed for some time before we became fully aware of it. We have enough Selectrons for the JOHNNIAC, but we are completely unhappy about the replacement problem. Very little is known about their life expectancy under dynamic conditions—JOHNNIAC JR. is just beginning a test program, so we will know something soon. Actually we never worried about this, partly because RCA initially indicated that the tubes would have life expectancies of at least several thousand hours—later tests (after we had received a good many tubes) shook our confidence—and partly because we expected the price to fall and the quality to improve so that eventually it would be OK whatever the initial conditions might be.

In passing: those of us responsible for the initial decision to go to Selectrons still feel that we used what brains God gave us in a pretty sensible way—though we deplore the result. The only remorse is that (for economy!) we decided not to keep the Williams tube as an insurance policy, i.e., as a parallel development.

We may be incurable optimists, but we think we see our way out of this mess, covered with diamonds moreover. We considered, briefly, returning to Williams tubes. For emotional reasons (and a couple of engineering ones!), we would just about as soon see JOHNNIAC chopped into those little pieces and put down the drain as to see it hooked up to this degrading companion at this late date. Our immediate plan is to go ahead with the Selectrons (we will soon own 100), as an interim measure. We may put JOHNNIAC on a reduced diet—256 words (40 tubes) instead of 512 words (80 tubes)—if the life tests on JOHNNIAC JR. make this appear advisable. (We can do a lot of computing with 256 words. In fact, if really pressed, we can drop to 20 tubes and 128 words.) Our aim is to get at least a year of work out of the Selectron memory.

The sunshine, which is giving us that exalted look again, is attributable to the little ceramic doughnuts—the magnetic cores. These have been on the horizon for several years and have looked like the next generation of high speed memory—completely in the spirit of the Selectron. logically, but simple and practically permanent. They looked wonderful whenever someone succeeded in mixing the right kind of mud, but when they did succeed they didn't know what they had done right. Within the last year at least two organizations—RCA and General Ceramics (and IBM is beginning to catch on)—have learned how to do it, dropping the magic wand only fitfully now; you can buy them today just as you'd buy rivets, and groups all over are making lattices and trying to build things out of them. There are some problems, but the general feeling is that any competent group can lick them in one way or another. A 1024-word memory is now running at MIT; it has a few bugs, but looks better than Williams after only one month of shake down. Our own group is confident of its ability to build one—and would dearly love to do it—and we may end up doing it ourselves. However, time is worth something to us and we would have to schedule it for later, owing to the present commitments of our small group. So if we can get it done reasonably on the outside, we would rather buy it. We sent detailed specifications and invitations to bid to about a dozen and a half vendors a couple of weeks ago. Seven have expressed a desire to bid—International Telemeter, Brush, Eckert-Mauchly (Remington Rand affiliate), RCA, IBM, Bendix, and Magnetic Research. We shall know more about this very soon. In any event, we feel off the hook memory-wise. Mark I will have Selectrons; Mark I Mod I will have magnetic cores, probably a year later.

Where do we stand? We have, since November 1950, spent about $200,000 for materials and outside work, and about $150,000 in salaries, including a fair pro rata share of departmental supervisory salaries (why should I be so fair?). If you wanted to be real dirty, you could add another $110,000 to represent 75% overhead on the salaries—but if you did, I might be goaded into mentioning hidden extras, such as consulting on RAND hardware problems, maintenance of electrical equipment throughout the RAND buildings, assistance to the Systems Research Laboratory, and planning for the move to the new building, and the lock on your office door. Also about $10,000 for travel.

We have quite a lot to show for it, and more that will show. We have JOHNNIAC JR., which is a faithful fraction (one-fourth) of Senior, with arithmetic unit, Selectron memory, and complicated elements of control. Many bugs, big and little, have been worked out here and will never plague Senior. Senior's arithmetic unit can be completed in short order whenever we throw our manpower on it—present schedule calls for arithmetic unit tests in about August. The main frame is completed and 34 of the eventual 42 registers are complete; of the remaining eight, four are in process and four (one-half full units, part of memory) await Junior's verdict. The main case is designed and prototyped and the most messy parts (panels for doors on end boxes) have been built. The Selectron memory is designed, and prototyped in Junior; Senior construction will go fast if Junior's tests are satisfactory. The air conditioning system is almost installed (about two weeks to go), as is a large motor generator set and a stand-by. Most of the power supplies have been installed. As you can see, we have spent a smaller part on the showy gadget itself. Despite our semileisurely pace, we don't look too bad on this list:

A Recap of the RAND Cost for JOHNNIAC

Purchases	$200,000
Salaries	150,000
Travel	10,000
Department Cost	$360,000
Overhead	110,000
RAND Cost	$470,000

Estimated Construction Periods for Princeton Machines and Copies

1. IAS: June 1946 to about October 1952 — $6\frac{1}{3}$ years
2. Los Alamos: July 1949–January 1952 — $2\frac{1}{2}$ years
3. University of Illinois: July 1949–March 1952 — $2\frac{2}{3}$ years
4. Argonne: September 1949–January 1953 (two machines) — $3\frac{1}{3}$ years
5. RAND: November 1950–? — $2\frac{2}{3}$ years to date

On the above completion dates, these were all barebones machines—Williams tubes plus teleprinter input-output plus nothing; most of them have since added (or are adding) drums and cards (switch from teleprinter). We don't know what the dollar costs were at the other institutions. It is evident, however, that our costs in money and elapsed time will be inflated by the Selectron interlude—perhaps by 12-18 months

and by $200,000-$300,000—if we measure history from the moment when we have a machine which we regard as being sound as a dollar (in the old sense!); i.e., one based on cores, big drum, and card input-output. This grim circumstance is mitigated more than somewhat by two factors: First, unless Nature is *very* unkind, the Selectrons will do a lot of computing for us during their year; second, it will take at least as much (in fact, more) additional time and money to bring the other Princeton-type machines up to a comparable level of modernization. (The transition Williams→ cores is much harder than Selectrons→ cores, requiring extensive rework; and of course the poor guys don't have room in which to do all this stuff—maybe they *can't* be modernized to JOHNNIAC level; to hear me talk, you'd never believe that all we have is an assortment of pieces! Ah, me...) So another way to look at our trouble is to say that we have tricked ourselves into building a better Mark I machine than we had planned, so perhaps the only lasting damage will be that suffered by our pride.

We will probably pay out another $50,000 of department funds before reaching the prime-number stage, what with bills for Selectrons and air conditioning still coming in. After that it may require about $75,000 to get the first really important JOHNNIAC. The major item needed which is not in the works now, hardwarewise, is a big drum. Much of the control for the drum has been designed and we are about ready to place an order for a good one—about $35,000. The small drum ($5,000) has given us some needed information and experience, but, since we are not going to tapes in Mark I and will have a relatively small high-speed memory initially, we need a larger intermediate-speed memory. Incidentally, ERA seems to have learned the drum business and purchasers are very pleased. IBM decided to build its own, and this is the best current headache on the 701. There is a console in the works, too, which is lagging for lack of a free engineer; the console is not essential, but not very costly and very convenient.

Our first working version of JOHNNIAC—Selectrons plus drum card input-output (hooking up a standard IBM machine)—will thus cost about as much as 701 rental for five or six quarters. It will be less flexible than the 701, because of the lack of tapes and smaller high-speed memory, but somewhat faster inside and more reliable. It will be very efficient for problems which have small input, which blow up into a lot of computing in the middle, and which boil down to a small output—a general characteristic of scientific problems, as opposed to census-type and data-handling problems.

If the money is available, we can probably usefully spend from $100,000 to $250,000 a year for several years—embellishing this machine, making a copy, or possibly going to transistors (if vendors are no more available then than they are for Princeton-types now); some embellishment (like magnetic cores) is essential, and tapes would be nice. But why bother? Why not just rent a 701? It's such a silly question I'm embarrassed for having written it. It's like questioning the economics of owning a Cadillac, on the one hand, and an oil well on the other. The 701 will only work if you have a nickel to put in it (some nickel), whereas the JOHNNIAC will do hundreds of thousands of dollars worth of computing each year as long as you pay the light bill. Actually each needs a staff; the real cash outlay to operate a 701 is probably like $600,000 a year and that of a comparable JOHNNIAC about $250,000 a year. It's like the REAC; we probably have $200,000 invested there, yet it will do prodigious work now for a song.

Incidentally, obsolescence is not a very important matter. While we will always want and be able to use better machines as they become available, a machine like the JOHNNIAC will always be a fine and useful machine—it will be able to solve just as many and as difficult problems ten years from now as it will next year; the log log duplex slide rule is still after fifty years the best way to evaluate a few hundred three-digit products and exponentials, and my twenty-year-old machine is about 95 percent as efficient as a brand new one; also note the REAC.

So, kindly cease asking silly questions (except of me) which only serve to make our crew feel insecure and unloved. The odds are very attractive that we will end up with a gorgeous useful gadget; and that the members of our computer group will have the best reputations in the country. But I don't believe anyone in my crowd will really begin to enjoy life until those Selectrons recede into history.

If you ever want another brief note on this subject, feel free to call on me. □

Meetings in Retrospect

Washington Computer Reminiscences

By Virginia C. Walker

The Computer Reminiscences (pre '54) Federal Involvement Conference, focusing on the years prior to 1954, was held in the Carmichael Auditorium of the National Museum of History and Technology, The Smithsonian Institution, on October 23, 1978. Co-sponsored by the Smithsonian and the D.C. Chapters of the Association for Computing Machinery, the Data Processing Management Association, and the IEEE Computer Society, this conference was conceived as a way to bring to the Washington area ADP community an overview of the early days of computing and an insight into the high level of federal involvement in the development of the earliest commercial computer systems. In addition, the conference was seen as an opportunity to gather together colleagues in these early experiences to share the oral history of the period and to expand the body of knowledge surrounding the events on which today's technology is based.

Members of the Steering Committee for the conference were: Co-Chairmen: Dr. Uta C. Merzbach, The Smithsonian Institution, and Solomon Rosenthal, Department of the Air Force; Arrangements: Virginia C. Walker, Department of Energy; Program: Joseph H. Easley, Environmental Protection Agency; Publications: Michael J. Healy, Better Impressions Incorporated; Registration: Betty J. Stevens, Department of Housing and Urban Development; Secretary: Robin L. Willingham, Department of the Air Force; Treasurer: William P. LaPlant, Jr., Department of the Air Force; Liaison to Local Chapters: William P. LaPlant, Jr. to ACM, William K. Krutz to DPMA, and Marvin V. Zelkowitz to IEEE Computer Society. Advisors to the Steering Committee were: Margaret R. Fox, Consultant; Dr. Carl Hammer, UNIVAC; and Captain Grace M. Hopper, USNR, Department of the Navy.

It was a successful conference if measured by the objectives stated above, and very successful in the eyes of the participants. One's first and continuing impression throughout the day was that all speakers were not only pleased to be able to relate their involvement in these pioneer experiences but were also overjoyed to see so many friends and peers. The general ambiance was dampened only when word was received that Dr. John and Mrs. Kathleen Mauchly, Dr. James MacPherson, and Ida Rhodes were unable to attend the meeting, although each sent greetings.

The day began with the showing of historic film sequences for early arrivals. Then Co-Chairman Rosenthal opened the conference with introductions of the Steering Committee and of the Honorable Michael Collins, Under Secretary of the Smithsonian Institution, who gave a brief welcoming address.

Program Chairman Joe Easley then presented Ethel C. Marden who moderated the first session entitled Early Concepts of Computer Technology. Other participants in this session, which focused on the evolution of computer technology in the first commercially available computer systems, were Dorothy P. Armstrong, Dr. Joseph Blum, Sid Greenwald, and Arthur W. Holt.

The second session, Programming Evolution, was moderated by Dr. Franz L. Alt; it covered design and development of early applications programs and software systems using machine language. Panelists were Dr. Elizabeth Cuthill, Frances E. Holberton, Captain Grace M. Hopper, Walter Jacobs, Dr. Eli Marks, and Nora Taylor. More historic film sequences were available for viewing during the lunch break.

The first afternoon session on Systems Development of Hardware and Software was moderated by Edward Dunaway. This session featured Dr. Murray Geisler, Morris H. Hansen, Edward S. Stein, Jacob Rabinow, and Lou Wilson, who discussed the developments leading to systems with faster processors, more internal storage and registers, better peripheral devices, and faster I/O.

The next session was organized by Margaret R. Fox on the UNIVAC I, The First Government Installations. Persons connected with these early installations were Donald H. Heiser, Emil D. Schell, John W. H. Spencer, and Dr. John W. Wrench. They compared their experiences and insights during these pioneering years.

Richard Sprague was the moderator of the session on Market Development in the Washington Area—Commercial and Governmental. Discussants Richard M. Bloch, Fred Ikenberry, and A. J. Neumann looked at the developing market for computing equipment as it was in the early fifties, commenting from the perspective of the vendor.

The final session, Computer Reminiscences—A View Toward the Future, was moderated by Samuel S. Snyder. Panelists Walter L. Anderson, Ray L. Bowman, and Robert P. Stephens discussed the potential use of computers for solving

Santesmases, J. G. 1962. Organizer of the Symposium on Advanced Methods in Information Storage and Retrieval. Munich, Proc. IFIP Congress.

Santesmases, J. G. 1963*a*. Non-linear resonance switching devices. *Switching Theory in Space Technology*, H. H. Aiken and W. F. Main (eds.), Stanford, Stanford University Press.

Santesmases, J. G. 1963*b*. *Las calculadoras electrónicas y la Universidad*. (Discurso correspondiente a la solemne apertura del curso académico 1963–64 en la Universidad de Madrid.) Madrid, Artes Gráficas.

Santesmases, J. G. 1965*a*. Data Processing in Spain. *Data Processing Magazine*, May, pp. 28–29.

Santesmases, J. G. 1965*b*. Presente y futuro de la automática. *Las Ciencias*, Madrid, Vol. XXX, No. 1.

Santesmases, J. G. 1974. Ferroresonant devices. In *Trends in Control Components*, M. Nalecz (ed.), IFAC Monographs, Amsterdam, North Holland Pub. Co., pp. 155–186.

Santesmases, J. G., M. R. Vidal, and J. Sánchez. 1954. Circuito disparador basado en la ferrorresonancia en paralelo. II. Uso de ferritas. *Anal. Real Soc. Esp. Fis. y Quim. 50-A*, Madrid, p. 47.

Santesmases, J. G., J. Sánchez, and J. Miró. 1956. Contribución al estudio de los circuitos de conmutación con rectificadores de Selenio. *Anal. Real Soc. Esp. Fís. y Quim. 52-A*, Madrid, pp. 105–116.

Santesmases, J. G., M. Alique, and J. L. Lloret. 1960. Ferroresonant Systems of Circuit Logic. *Proc. IEE London*, Vol. 107, B, No. 32, pp. 190–198.

Santesmases, J. G., J. L. Lloret, and J. Ayala. 1966. Expresiones analíticas para ciclos de histéresis. *Anal. Real Soc. Esp. Fis. y Quim. 62-A*, Madrid, pp. 217–222.

Santesmases, J. G., J. Ayala, and A. H. Cachero. 1966. Dispositivos de medida de ciclos dinámicos de histéresis punto a punto. *Ciencia Aplicada*, Madrid.

Santesmases, J. G., J. Ayala, and A. H. Cachero. 1967. Analog simulation of a ferroresonant system including analysis of hysteresis loop. *Proc. Intl. Association for Analog Computation*, No. 2, pp. 1–5.

Santesmases, J. G., J. Ayala, and A. H. Cachero. 1968. Estudio analítico de sistemas ferrorresonantes con inclusión de la histéresis. *Anal. Real Soc. Esp. Fis. y Quim. 64-A*, Madrid, pp. 131–139.

Santesmases, J. G., J. Ayala, and A. H. Cachero. 1970. Analytical Approximation of Dynamic Hysteresis Loop and its Application to the Ferroresonant Circuit. *Proc. IEE London*, Vol. 117, No. 1, pp. 234–240.

Santesmases, J. G., J. Solé, A. Vaquero, and J. M. Guillén. 1969*a*. Realización de un sistema de enseñanza automática (Part I). *Revista de Automática*, Madrid, No. 3.

Santesmases, J. G., J. Solé, A. Vaquero, and J. M. Guillén. 1969*b*. Realización de un sistema de enseñanza automática (Part II). *Revista de Automática*, Madrid, No. 4.

Santesmases, J. G., E. Luque, and J. M. De la Horra. 1974. Miniordenador IEA-FI. *Revista de Informática y Automática*, No. 19, Madrid, pp. 5–13.

Neuron Networks and Learning Systems

Moreno Diaz, R., and J. Bever. 1967. Visual data processing. In *Sensory, Decision and Control Systems*. L. Sutro (ed.), R-565, Cambridge, MIT Press, pp. 32–36.

Moreno Diaz, R. 1967. Visual processing in animals. In *Sensory, Decision and Control Systems*. L. Sutro (ed.), R-565, Cambridge, MIT Press, pp. 62–65.

Rubio, F. 1969. *Modelo de procesos retinales. Simulación electrónica y en calculadora*. Ph.D. Thesis. Universidad Complutense, Madrid.

Santesmases, J. G. 1962. Sistemas de inteligencia artificial. *Las Ciencias*, Vol. XXVII, No. 2, Madrid.

Santesmases, J. G. 1965. Biónica: nueva frontera del conocimiento. *Revista de Ciencia Aplicada*, No. 107, Madrid, p. 481.

Santesmases, J. G. 1970. *Cibernética y proceso de la información visual en los seres vivos y en las máquinas*. Madrid, Real Academia de Ciencias Exactas, Fisicas y Naturales.

Santesmases, J. G. 1974. *Instrumentación electrónica para el estudio y simulación de funciones globales del sistema nervioso*. Madrid, Fundación Juan March.

Santesmases, J. G., and R. Moreno Diaz. 1963. Algunas analogías electrónicas de elementos neuronales. *Anal. Real Soc. Esp. Fis. y Quim. 59-A*, Madrid, p. 251.

Santesmases, J. G., and R. Moreno Diaz. 1964. Investigaciones sobre modelos neuronales. *Congreso Luso-Español para el Progreso de las Ciencias*, Bilbao.

Santesmases, J. G., and R. Moreno Diaz. 1965. Neuronas artificiales para la simulación del reflejo condicionado. *Anal. Real Soc. Esp. Fís. y Quim. 61-A*, Madrid, p. 3.

Santesmases, J. G., and F. Rubio. 1968. Modelo de inhibición lateral en el ojo compuesto del Limulus. *Anal. Real Soc. Esp. Fis. y Quim. 64-A*, Madrid, pp. 243–250.

Santesmases, J. G., F. Rubio, L. Arés, and R. H. Verduzco. 1969. Harmonic Analysis in the Visual Pathway of the Higher Mammals. *Proc. Intl. Congress of Cybernetics*, London, Vol. I, Chaps. 11–13, pp. 403–417.

International Activities

IEA. 1958. Congreso Internacional de Automática (International Congress of Automatics), Madrid, CSIC.

Naslin, P., 1959. Le Congrès International d'Automatique de Madrid. *Automatisme*. Vol. 4, No. 4, pp. 18–37.

Revista de Informática y Automática. Facultad de Ciencias Fisicas, Ciudad Universitaria, Madrid.

JOSS—Conversational Computing for the Nonprogrammer

SHIRLEY L. MARKS

JOSS, the JOHNNIAC Open-Shop System, is a conversational time-sharing system developed at the Rand Corporation to demonstrate, on a small scale, the value of time-sharing and easy access to computing power for the nonprogrammer. In three computer implementations from May 1963 to the 1980s, JOSS has provided an English-like language that is easy to learn and use by trial and error at a terminal. Thus this earliest of simple on-line systems has enabled the computer novice to explore to advantage many small computational problems that might not be worth the effort in another computing environment.

Categories and Subject Descriptors: D.3.2 ***[Programming Languages]:*** *Language Classifications;* K.2 ***[History of Computing]***—software, JOSS; hardware, JOHNNIAC
General Terms: Human Factors, Languages
Additional Key Words and Phrases: time-sharing

1. Implementing an Ideal

Shortly after the Rand Corporation moved to its present location in Santa Monica, California, in early 1953, the JOHNNIAC was completed and installed as Rand's first stored-program computer. Eight years later, this Princeton-type computing machine (named for mathematician John von Neumann) became the basis for JOSS,[1] the JOHNNIAC Open-Shop System.

The first truly simple on-line system, JOSS represents a milestone in the history of conversational time-sharing. In the years of its operation at Rand (1963 to the 1980s), JOSS has more than lived up to its billing as "The Helpful Assistant." But more significant than its long service to the Rand staff has been its influence on interactive system design.

JOSS was the result of an experimental project at Rand that was meant to demonstrate, on a small scale, the value of time-sharing and easy access to computing power for the nonprogrammer. Although few of today's time-sharing users may know its name, many are benefiting from JOSS concepts of user-oriented language and terminal. These concepts had their origin in the 1950s, when the electronic computer was transformed from a one-of-a-kind laboratory curiosity into a marketable commodity.

As new applications were promoted, the language of computer instruction sets rose from the level of machine code to that of assemblers and compilers. At the same time, professional programmers were replacing those pioneering scientists and engineers who had learned to cope with the complexities of the machine in order to use this remarkable tool. Formerly a researcher seated at the computer's console had had its immediate and undivided attention. Now such dedicated personal usage was generally an unaffordable luxury.

But while the problem originators were moving farther away from their electronic problem-solvers, others were moving closer. For the 1950s also saw the first large-scale real-time use of computers in the SAGE system, developed for air defense command and control (Sackman 1970). The SAGE operators seated at

Author's Address: 2873 Globe Avenue, Thousand Oaks, CA 91360.
[1] JOSS is the trademark and service mark of the Rand Corporation for its computer program and services using this program.
© 1982 AFIPS 0164-1239/82/010035-052$01.00/00

their many terminals experienced a new relation between humans and machine—a simultaneous sharing of the computer's resources.

At the Rand Corporation, Cliff Shaw envisioned the possibilities for a particular type of user not likely to be satisfied by the prospective commercial or governmental systems. In 1955 Cliff expressed his concern for the Rand staff's "open-shopper,"[2] the nonprogrammer physicist, engineer, economist, or mathematician with a small numerical problem. Sometimes such a problem required only the features of the desk calculator of that day (which was far more elementary than today's hand-held calculator). At other times the sophisticated capabilities of an electronic computer were needed. Cliff was also aware of the researcher's occasional desire to intervene in the calculation process.

A 1959 memo from Rand engineer Willis Ware suggested that future information processors would require "a multiplicity of personal input-output stations, so that many people can interact with the machine at the same time." In November 1960, meetings were held at Rand to decide the future of the JOHNNIAC.[3] Attending were Paul Armer, head of the Computer Sciences Department, Willis Ware, associate department head, engineers Keith Uncapher and Tom Ellis, and research programmer Cliff Shaw.

Cliff recommended that the JOHNNIAC be used full time to service the open shop by means of hard-copy stations, and he looked forward to the challenge of resolving communication and monitoring difficulties. The JOHNNIAC, in its youth, had served well as a production machine. When its storage and speed limitations eventually caused a shift to more modern equipment, this aging home-grown computer became attractive for the JOSS experiment.

There have been many creative contributors to JOSS's evolution from experimental to operational system (see the Appendix for the JOSS "Cast of Characters"). Among them, Cliff Shaw set out to establish the JOSS precept of "easy to learn and easy to use" as a principle to govern the entire system design. As Cliff later recalled, his intent was not to make JOHNNIAC machine language available, but instead to provide a computational service through a new, machine-independent language. The system would be designed specifically to show the value of on-line access to a computer via a language tailored to a special class of user (the nonprogrammer) and a special type of application (small and numerical).

It was in March 1961 that JOSS was formally proposed as "an exploration into continuous and intimate contact between a human user and a computer." So began the first phase of the U.S. Air Force-sponsored Information Processor Project, whose goal was to improve communication between human and machine. An unsung hero of the project, recalls Chuck Baker who directed the second phase, was John Williams, the head of Rand's Mathematics Department. John, according to Chuck, "backed JOSS administratively in his own quiet but powerful way. His influence was vital when it came to getting the funds for the work. His encouragement and enthusiasm helped us all over some very rough spots, too. And he had a lot of influence in what JOSS I finally looked like." John Williams, along with Norm Shapiro, Art Smith, Oliver Gross, Bill Sibley, and others, provided valuable feedback to designers of JOSS I and II.

2. From JOSS I to JOSS III

Participants with Cliff Shaw in project discussions were Tom Ellis, Ike Nehama, Al Newell, and Keith Uncapher. Included in their plans were 10 typewriter terminals in fixed locations. Tom Ellis and Mal Davis directed the construction of the required multiple-typewriter communication system during 1961 and 1962. Meanwhile Cliff was designing the interdependent details of language, terminal, and conversational environment.

JOSS was inaugurated in May 1963 with an initial five terminals and a minimal system. One terminal was installed at the JOHNNIAC, and four were located in the offices of Rand staff selected to evaluate JOSS. There were two principal elements of the terminal, from the user's point of view. One was an IBM model 868 typewriter. The other was a small box that indicated the status of the terminal's communication electronics and controlled their functions. (See Table 1, "JOSS I Control Box," in Section 4.)

The first schedule of operation, 9 to 12 each weekday morning, was announced on June 17. It was hoped that this limited edition of the system could provide a controlled feedback that would help shape the complete version. But the new users taught others so quickly that the experimenters had to resort to after-the-fact questionnaires.

January 1964 marks the official birth of the full JOSS implementation on the JOHNNIAC. By July of that year, JOSS service was extended into the evening hours to accommodate Rand staffers' growing addiction. The final version of JOSS I went into use in January 1965. Jean Sammet, in her history of programming languages (1969), notes: "The most amazing aspect of the whole JOSS activity is that it provided such a useful tool to so many people at the Rand Corporation." "And continued to provide," might be added as we consider the progression to the 1980s version.

With the increasing usage of JOSS came the worry that the JOHNNIAC's shortcomings might prejudice the system's evaluation. The project group emphasized that an up-to-date computer teamed with numerous user-oriented terminals would advance its interactive character. The additional storage and speed would not only increase its power as a calculating aid and provide long-term storage for programs and data; these features would also enable JOSS to monitor its own performance. So in July 1964, anticipating the JOHNNIAC's retirement to the Los Angeles County Museum, a second JOSS was proposed by Chuck Baker.

Several computer equipment manufacturers were invited to submit bids for the new dedicated processor. Each of the nine bids received was rated on such factors as response time, arithmetic computation speed, ability to store internally an exact representation of user-typed input, inclusion of terminals built to Rand specifications, and date of delivery. Ratings at last completed, vendors for all system components were selected and Air Force funds were approved.

The contract signed with Digital Equipment Corporation promised installation of a PDP-6 computer by October 31, 1965. DEC would also build 30 terminals according to Rand requirements. Each terminal would include a modified IBM Selectric typewriter and a Rand-designed "paging" mechanism, a mechanical device that would permit the typewriter to space in a single motion to the top of the next fanfold page.

Data Products would supply a disk file for between-session storage of user programs. Vermont Research would furnish a magnetic drum for temporary storage of programs during sessions. James G. Robins Electronics agreed to build a line-finder, which would locate an available channel to the PDP-6 for each terminal connected to a special office outlet. Standard data communications equipment would couple JOSS to terminals outside the Rand buildings.

Following installation of a minimum configuration of the PDP-6 in late July 1965, the programming team began to check out the software by Teletype terminal. In charge of the JOSS II group was Chuck Baker, who contributed many human-engineering notions to the new terminal and was primarily responsible for the new features of the language. Joe Smith created the central processing routines, which interpret and respond to requests in the JOSS language. Irwin Greenwald programmed the arithmetic and function evaluation routines, as well as the software that handled the JOSS II terminal-computer communications and enabled the user to access files on disk. Ed Bryan designed the JOSS monitor, which assigns priorities to user requests and states, allocates resources, and schedules tasks. The monitor also contains routines to accumulate accounting information and record performance statistics.

By October 1965, checkout of the minimum JOSS II was finished. In November, a prototype JOSS terminal arrived from DEC. In December at the Fall Joint Computer Conference in Las Vegas, wives of attendees witnessed the first use of the system outside Rand when a friendly blue terminal typed "JOSS at your service. Initials please:" But the performance that delighted the wives with its appealing simplicity gave no hint of the unbelievable difficulties experienced by the JOSS group in the prior frantic hours. Obtaining modems and data lines involved negotiating with three telephone companies in two states. Actual data transmission was "iffy" until the final moments.

Chuck Baker wrote to users on February 9, 1966: "As all of you are by now aware, Friday, February 11 will be the last day on which JOSS service from JOHNNIAC will be available—'The end of an era.'" On February 18, Willis Ware (1966) delivered a eulogy to the JOHNNIAC before a gathering of old friends. Then Cliff Shaw loaded a final program that counted down the seconds to the JOHNNIAC's demise (Figure 1).

During the following months, the PDP-6 was moved into the Rand basement, production terminals were delivered, and the drum and disk joined the line-finder in the new system. JOSS II operational hours were lengthened as the system became more reliable. By the end of 1966, service was being provided 24 hours

---

Open shop is a well-known labor-management term denoting a worksite open to nonunion as well as union members. Thus an open-shop computer installation came to mean one available to nonprogrammer and professional programmer alike. A nonprogrammer user was called an "open-shopper."

In 1950 Rand needed to increase its computing power significantly over what it obtained from IBM Card-Programmed Calculators. After surveying commercial and university projects, Rand decided to build an improved version of the machine being constructed under the leadership of John von Neumann at the Institute for Advanced Study in Princeton. It was to he called JOHNNIAC. By the time Shaw was installing JOSS I on the JOHNNIAC, the machine had 4096 words (40 bits each) of core storage with a cycle time of 15 microseconds, drum storage of 12,288 words, punched-card input-output, and a high-speed printer. It had no magnetic tapes, no indexing, and no built-in floating point. For details of JOHNNIAC's life (1953–1966), see F. J. Gruenberger, "The History of the JOHNNIAC," *Annals of the History of Computing*, Volume 1, Number 1, pp. 49–64.

Shirley Marks worked at the Rand Corporation for 18 years, 14 of them as an applications programmer in the Computer Sciences Department and the last 4 as manager of JOSS II service. She has an M.A. in mathematics from UCLA, and is at present a consulting technical writer and freelance writer for general-readership publications.

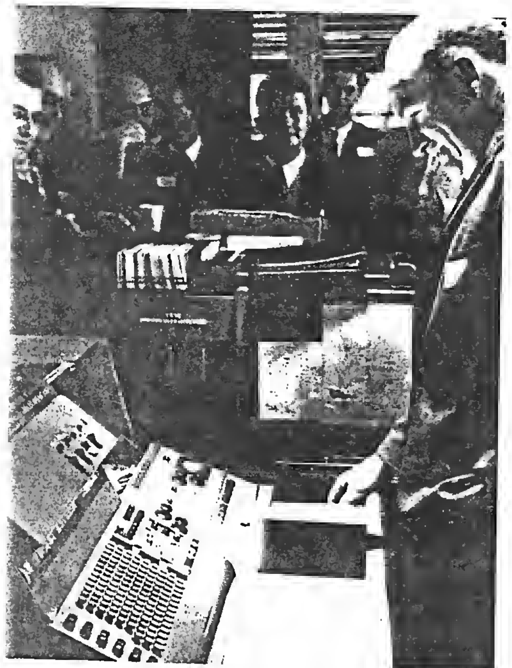


Figure 1. JOHNNIAC retirement, February 18, 1966. Cliff Shaw is at the console. In the background, left to right: I. D. Greenwald, J. W. Smith, unidentified, W. L. Sibley, G. E. Bryan, A. C. Smith.

a day, 7 days a week, minus hours for maintenance. In addition to the terminals at Rand, there were also initially dedicated communication lines to the AEC's Nevada test site and to various Air Force locations. Eventually both Rand and the Air Force made use of dial-up service from sites around the country.

PDP-6 JOSS, with thirty times the speed of JOHNNIAC JOSS, five times the storage, three times the number of terminals, and powerful new language features, had broadened Cliff Shaw's original design. But the basic principle of a handy computing tool for the computer novice had been preserved.

In April 1967, management of JOSS was transferred from the development group to a small staff under George Armerding. Larry Clark took charge of software upkeep. Art Lucero attended to hardware and communication needs. Shirley Marks offered aid and comfort to users.

It might be said that old software never dies—its hardware fades away. So it was with JOSS on the JOHNNIAC, and so also on the PDP-6, although not in the same dramatic style. This time there was no ceremonial retirement to a museum, only orderly removal of an outmoded computer in the summer of 1974. But JOSS had already made the transition to its new host, an IBM 370/158. Larry Clark, assisted by Donna Cooper, did the honors, installing a JOSS II based on a DOS version. (Earlier, a JOSS II replica called "MATH" had been created by John Disbrow at IBM's Advanced System Development Division in Los Gatos, California.) A preliminary version was running on April 15, 1974. A production form was in operation by July 1.

The main difference between JOSS II and JOSS III is the hardware setting. No longer does JOSS have exclusive tenancy of a computer, nor do users communicate at a unique JOSS terminal. Yet the essence of the system is the same as when it was conceived some 20 years ago. Part of that essence is apparent in the language.

3. The JOSS Language

At the 1964 Fall Joint Computer Conference in San Francisco, Cliff Shaw (1964) presented JOSS from a designer's point of view. Topping his list of characteristics were "readability of the language, the high degree of interaction, and the power of expression." More impressive to the JOSS I learner, however, was the obvious simplicity of Shaw's one-page summary of the language (Figure 2).

Readability is evident here in the use of full-word English-like sentences, which begin with a capitalized imperative verb and end with a period. The JOSS I novice, armed with a single page of commands, relations, conditions, and functions, could explore the system capabilities and restrictions by trial and error at a terminal. Succeeding generations of users have done the same with the extended JOSS II language (Figure 3).

Chuck Baker likes to compare exploring JOSS capabilities with discovering how to use his modern pocket calculator or even the old-fashioned desk calculator. In each case, the novice can try out the tool in private, making "stupid" mistakes no one else knows about and learning from those mistakes. In this way the user progresses at his or her own pace to more complex applications. Chuck also observes the similarity between several JOSS features and those of his pocket calculator and wonders if there is a historic line of descent.

The interactive character of JOSS rests in part on the interpretive nature of the language. There is never a "compilation" to produce an "object" program. Instead, a user statement is analyzed character by


```
DIRECT or INDIRECT

  Set x=a.

  Do step 1.1.
  Do step 1.1 for x = a, b, c(d)e.
  Do part 1.
  Do part 1 for x = a(b)c(d)e, f, g.

  Type a,b,c,_.
  Type a,b in form 2.
  Type "ABCDE".
  Type step 1.1.
  Type part 1.
  Type form 2.
  Type all steps.
  Type all parts.
  Type all forms.
  Type all values.
  Type all.
  Type size.
  Type time.
  Type users.

  Delete x,y.
  Delete all values.

  Line

  Page.

INDIRECT (only):

  1.1 To step 3.1.
  1.1 To part 3.

  1.1 Done.

  1.1 Stop.

  1.1 Demand x.

DIRECT (only):

  Cancel.

  Delete step 1.1.
  Delete part 1.
  Delete form 2.
  Delete all steps.
  Delete all parts.
  Delete all forms.
  Delete all.

  Go.

  Form 2:
  dist. = .........  accel. = __.___

  x=a

RELATIONS:

  = ≠ ≤ ≥ < >

OPERATIONS:

  + - · / * () [] ||

CONDITIONS:

  if a<b<c and d=e or f≠g

FUNCTIONS:

  sqrt(a)     (square root)
  log(a)      (natural logarithm)
  exp(a)
  sin(a)
  cos(a)
  arg(a,b)    (argument of point [a,b])
  ip(a)       (integer part)
  fp(a)       (fraction part)
  dp(a)       (digit part)
  xp(a)       (exponent part)
  sgn(a)      (sign)
  max(a,b)
  min(a,b,c)

PUNCTUATION and SPECIAL CHARACTERS:

  . , ; : ' " _ # $ ?
  ___.____ indicates a field for a number in a form.
  ......... indicates scientific notation in a form.
  # is the strike-out symbol.
  $ carries the value of the current line number.
  * at the beginning or end kills an instruction line.
  Brackets may be used above in place of parentheses.
  Indexed letters (e.g., v(a), w[a,b]) may be used above in place of x, y.
  Arbitrary expressions (e.g. 3·[sin(2·p+3)-q]+r ) may be used above
    in place of a, b, c, ....
```

Figure 2. JOSS one-page summary.

In the following list of commands and functions, these symbols are used:

L = letter
S = subscripted letter
P = proposition
F = formula

int = expression with integer value
num = expression with numerical value
rng = range of values of a letter, such as *1(.1)3,3.64,4*

In JOSS, an expression, e.g., *3+sqrt(2)*, has a numerical value. A proposition, e.g., *x≤5*, has a truth value (true or false). An *if* clause can be appended to any command except a short *Set* command. A conditional expression may be used wherever an expression is allowed; e.g., define f(x) by a conditional expression: *Let f(x)=(x<0:0;0≤x<1:x*2;1)* defines f(x) to be 0 for x<0, x*2 for 0≤x<1, and otherwise 1.

JOSS II Command List

TYPE COMMANDS
Type num.
Type S.
Type S(int,int).
Type P.
Type "any text".
Type _.
Type form int.
Type step num.
Type part int.
Type formula F.
Type F(num).
Type F(P).
Type all steps.
Type all parts.
Type all formulas.
Type all forms.
Type all values.
Type all.
Type time.
Type timer.
Type size.
Type users.
Type item-list.

- Several individual *Type* commands may be combined in one, except for *Type "any text"* and *Type item-list*.
- The words *in form* may be appended to *Type* commands that specify individual values.

DELETE COMMANDS
Delete L.
Delete S.
Delete S(int,int).
Delete form int.
Delete step num.
Delete part int.
Delete formula F.
Delete all steps.
Delete all parts.
Delete all formulas.
Delete all forms.
Delete all values.
Delete all.

- Several individual *Delete* commands may be combined in one, such as *Delete* L, *form* int, *all parts*.

SINGLE-WORD COMMANDS
Page.
Line.
Go.
Stop.
Done.
Quit.
Cancel.

Figure 3. JOSS II commands and functions.

SET COMMANDS
Set L=num.
Set L=P.
Set S(int,int)=num.
Set S(int,int)=P.

SHORT SET COMMANDS
L=num
L=P
S(int,int)=num
S(int,int)=P

LET COMMANDS
Let F=num.
Let F=P.
Let F(L)=num.
Let F(L)=P.

DO COMMANDS
Do step num.
Do step num, int times
Do step num for L=rng.
Do part int.
Do part int, int times.
Do part int for L=rng.

TO COMMANDS
To step num.
To part int.

PARENTHETICAL COMMANDS
Cancel or any *Do* command may be enclosed in parentheses.

SPECIAL COMMANDS
Reset timer.
Let S be sparse.

DEMAND COMMANDS
Demand L.
Demand S(int,int).
Demand L as "any text".
Demand S(int,int) as "any text".

FILE COMMANDS
Use file int (ident).
Use library int.
File ... as item int (code).
Recall item int (code).
Discard item int (code).

- The file number and identification are assigned by the Information Sciences Department.
- Library files are numbered from 1 to 250 and do not have an identification.
- The item number and code are assigned by the user at time of filing. 1≤int≤25. The code, if used, contains at most 5 letters and numbers.
- Any combination of elements that can be deleted by a *Delete* command may be filed by a *File* command.

Figure 3. *Continued.*

JOSS II Function List

```
                          ip(x) = integer portion of x
                          fp(x) = fraction portion of x
                          dp(x) = digit portion of x
                          xp(x) = exponent portion of x
                         sgn(x) = -1 if x<0, 0 if x=0, +1 if x>0

                        sqrt(x) = square root of x for x≥0
                         log(x) = natural logarithm of x for x>0
                         exp(x) = e^x

                         sin(x) = sine of x, x in radians, |x|<100
                         cos(x) = cosine of x, x in radians, |x|<100
                       arg(x,y) = angle in radians formed by the positive
                                  x-axis and the line from the origin to (x,y)

  sum(x,y,z) or sum[i=rng:num]  = sum of values in argument list
prod(x,y,z) or prod[i=rng:num]  = product of values in argument list

  min(x,y,x) or min[i=rng:num]  = minimum of values in argument list
  max(x,y,z) or max[i=rng:num]  = maximum of values in argument list

  conj(p,q,r) or conj[i=rng:P]  = true if and only if P is true for all i
  disj(p,q,r) or disj[i=rng:P]  = true if any P is true

                 first[i=rng:P] = first i for which P is true

                          tv(P) = the truth value of P; 0 if P is false,
                                  1 if P is true

      • num or P is evaluated for each value of i in the range
        rng; i usually appears in num or P.
```

Figure 3. *Continued.*

character (by the JOSS interpreter as well as by the arithmetic and function subroutines) each time the statement is encountered.

Internally, user commands and data are carried in linked lists within variable-size user blocks. The entire user block must be in high-speed memory in contiguous cells during interpretation. Whenever interpretation of a command is aborted for any reason, JOSS makes sure that all information stored away for the user is precisely as it was when JOSS began interpreting the command. This software technique enables the user to create or edit a stored program whenever control is returned, whether at the user's request or by an error, and then to continue.

Cliff pointed out in his 1964 paper that JOSS is commanded "directly" in the same language in which procedures are defined to be carried out later ("indirectly"). So there is no need to switch between direct and indirect modes. Merely by prefixing a one-line step with a numeric label, the user indicates that the step is to be stored rather than directly executed.

Steps whose labels have the same integer portion constitute a "part." Although the user may create steps (and parts) in any order, JOSS sorts steps within a part according to the fraction portion of the label (e.g., 1.1, 1.2, 1.3, 2.01, 2.1). Thus the choice of label determines if the step is an addition (1.4), insertion (1.15), or replacement (2.1). Steps and parts can also be displayed (e.g., "Type part 2, step 1.1.") or removed (e.g., "Delete step 1.2."), as can values, vectors, arrays, formulas, and format statements called "forms."

JOSS can be commanded to execute a single step or a part, once, repeatedly, for a range of values, as well as conditionally—yet within the length of a single line. Having received control, the user can make changes to program elements, display items of interest, and

directly command JOSS to resume. In the case where a stored step is repeated after an error interruption, JOSS will reinterpret the step in its entirety.

For ease of interaction, it is critically important that the user be made clearly aware of the status of JOSS whenever it is time to type. To this end there are error messages, interrupt messages, and stop messages to distinguish these states from a successful task completion. Error messages report exactly violations of language rules (e.g., "Error at step 1.1: I have a negative argument for square root.") with one exception. JOSS types the laconic "Eh?" when a malformed line is easier to analyze by eye than by software diagnosis (e.g., "Do paart 1.").

If JOSS runs out of space in performing a task, the user is so informed and given the opportunity to delete nonessential portions of the program. "Go." then instructs JOSS to carry on as before, but now with additional workspace. In this and in other aspects of interaction, the user can employ the language at a terminal to develop the program with a minimum of presession planning. Furthermore, the user never has to consult with JOSS system experts, or be knowledgeable in the ways of the computer, the software, or the internal look of the program or data. Of course, there are always other JOSS users willing to share techniques and personal experiences.

Considerable computational power is available within the single-line limits of a command. For instance, besides the more traditional statement types, there is the JOSS II "Let" command, which defines a formula whose letter name may then be used in subsequent commands. Up to 10 parameters are allowed. Then there are such functions as "sum," "prod," "max," and "min," which eliminate the need for many program loops while permitting the user to represent mathematical relations in a familiar way.

Expressions may be elementary or complex as desired, with numeric or true-false values, and may be used in place of values everywhere except as a step-label prefix. Expressions may also be conditional. For example, the formula $f(x)$ might be defined by an expression that is conditionally zero, x-squared, or one, depending on the applicable range of x values.

Let f(x) = (x < 0:0; 0 ≤ x ≤ 1:x**2; 1).

Note that the statement reads naturally from left to right.[4] JOSS interprets the conditions in that order, too. If the first condition holds ($x < 0$), JOSS evaluates the associated definition (0) and ignores the remainder of the expression, including its errors, and so forth. If none of the conditions hold and there is no default provided, an error results. In this example, "1" is the default definition for $f(x)$. If it had been forgotten and during execution x took a value of, say, 3, JOSS would respond with

Error in formula f: (x < 0:0; 0 ≤ x ≤ 1:x**2) = ???

Recursion can be expressed conveniently too, although such formulations often make greater demands on execution space and time than do iterative ones. A simple comparison is in the calculation of factorial n, which can be defined iteratively by means of the "prod" function.

Let F(n) = [n = 0:1; prod(i = 1(1)n: i)].

This can be read, "Define factorial n to be, for $n = 0$, 1; otherwise, the product, for i equal to 1 at steps of 1 to n, of i." The formula might be defined recursively.

Let F(n) = [n = 0:1; n*F(n-1)].

Here also there are the two cases. "Factorial 0" is defined to be 1; otherwise "factorial n" is calculated as the product of n times $F(n\text{-}1)$. Clearly the initial case is essential. For had this condition been omitted, JOSS would try its best, then type

Revoked. I ran out of space (in formula F).

As may already be apparent in the examples, the only admissible identifiers for values are the 52 upper- and lowercase letters. These, in turn, can be indexed to form vectors and arrays, such as $x(i,j,k)$.

One design requirement specified that JOSS should give the same results for arithmetic operations as those produced by a desk calculator. JOSS improves the calculator's results by providing automatic rounding as well as scientific (floating-point) notation when two large numbers are multiplied. To accomplish this, JOSS numbers are represented internally as integers with associated powers of ten, and all arithmetic operations are performed using integer arithmetic. The values the user supplies are treated as nine exact decimal digits; the values JOSS returns are true nine-digit rounded results in the case of add, subtract, multiply, divide, and square root. Magnitudes greater than $9.99999999 \cdot 10^{99}$ result in an overflow message; magnitudes less than 10^{-99} are replaced by zero. (Note that the JOSS III factor is 10^{63} or 10^{-63}.) Moreover, function evaluations hit on the nose "magic values" such as sine of 90 degrees.

Input is "freeform"—that is, the user may type in any columns of the page. The amount of spacing between terms is generally the user's option where space permits. Users may insert parentheses and brackets to improve readability or simply to clarify

[4] The double asterisk indicating exponentiation is a JOSS III convention, as is the single asterisk for multiplication. JOSS I and JOSS II keyboard symbology is discussed in Section 4. The JOSS Terminal.

their intentions when they aren't sure how JOSS will evaluate an expression. When users prefer a formatted output instead of the standard one value per line, they have the entire line width to describe both the line's literal information and the fields for numeric answers.

At this point an example will show something of the flavor of a JOSS session—the first tries at program formulation, the helpful reminders of error, the improvements and retries, and the final success. The exercise here is to print a table of roots of the general quadratic equation $ax^2 + bx + c = 0$ for a set of values a, b, and c. In the example, the first character of a user-typed line is underscored to distinguish it from a JOSS-typed line, as is the JOSS III custom. JOSS I and II conversations were in green (for the user) and black (for JOSS).

Having logged on, the user begins and immediately catches a typing error, which is nullified by an asterisk before pressing the Return key. The user then starts afresh, first creating "part 1" to calculate and type the roots.

```
1.1 Sett *
1.1 Set x(1) = [−b+sqrt (b**2−4*a*c)]/(2*a).
1.2 Set x(2) = [−b−sqrt (b**2−4*a*c)]/(2*a).
1.3 Type x(1), x(2).
```

The user assigns values to coefficients and directly commands JOSS to perform part 1.

```
Set a=1.
Set b=−1.
Do part 1.
Error at step 1.1:   c = ???
```

JOSS reminds the user to assign a value to the third coefficient.

```
Set c=−6.
Go.
        x(1) =  3
        x(2) = −2
```

Next the user decides to introduce a print format by replacing step 1.3.

```
1.3 Type a, b, c, x(1), x(2) in form 1.
Do part 1.
Error at step 1.3:   I can't find the required form.
```

With polite prodding by JOSS, the user describes the format, which provides three integer digits for each coefficient, and one integer digit followed by four decimal places for each root. The user indicates this format by typing the desired number of underscores and decimal points where required.

```
Form 1:
a=___ b = ___ c = ___ roots: _.____ _.____
```

The user continues by typing

```
Go.
Error at step 1.3: I can't express value in your form.
```

Remembering that one root is negative, the user now provides two integer positions for each field to allow room for the possible minus sign.

```
Form 1:
a=___ b=___ c=___ roots: __.____ __.____
Go.
a=  1 b= −1 c= −6 roots:  3.0000   −2.0000
```

A fancier format seems in order, but this will require a fancier program. The user first defines a new "form 1" without the literal information, and beneath it a "form 2" (using form 1 as a guide) to produce a heading for the five columns.

```
Form 1:
___ ___ ___ __.____ __.____
Form 2:
  a   b   c  Root 1  Root 2
```

The user creates parts 2, 3, and 4 to prescribe ranges for the three coefficients and to print the desired table of roots, complete with heading and spacing between blocks of printout. Of course, the user may have a bit of trouble and need to display helpful information. But eventually the following may be what the user ends up with, in addition to part 1 and the two forms above (all displayed by the command "Type part 2, part 3, part 4."):

```
2.01 Line.
2.1 Do part 1 for c=−1(−1)−4.

3.1 Do part 2 for b=−1(−1)−2.

4.01 Page.
4.02 Type form 2.
4.1 Do part 3 for a=1(1)2.
```

To produce the table, the user types

```
Do part 4.
```

JOSS responds with a new page and then

```
a   b    c   Root 1Root 2
1  −1   −1   1.6180  −.6180
1  −1   −2   2.0000 −1.0000
1  −1   −3   2.3028 −1.3028
1  −1   −4   2.5616 −1.5616

1  −2   −1   2.4142  −.4142
1  −2   −2   2.7321  −.7321
1  −2   −3   3.0000 −1.0000
1  −2   −4   3.2361 −1.2361

2  −1   −1   1.0000  −.5000
I'm at step 1.3.
```

Satisfied that the program is running properly, the user presses the Interrupt key. JOSS prints a line or two more and then an "interrupt" message.

The user may now choose to modify the program further and resume, file it for later retrieval, or simply discard it as one might crumple a page of pencil figuring. Were this a JOSS II session, the user might also have swiveled the office chair from the terminal to the desk, or unplugged the terminal to wheel it to another user's office. This mobility of the JOSS II terminal was only one facet of its contribution to the interactive JOSS environment.

4. The JOSS Terminal

The "face" of JOSS, all that most JOSS I or JOSS II users ever saw, was a familiar-looking electric typewriter, to the right of which a small box displayed a few buttons, switches, and lights (Figure 4). Unintimidated by the terminal's appearance, the beginner felt little hesitancy in pressing a switch to "on." Users would soon discover if they had turned on only the typewriter, which was available for off-line typing when not online to the system. If, instead, they pressed a control-box power switch, there would be the JOSS greeting, then lights and a soft sound notifying them it was their turn to type; the two-color conversation was under way.

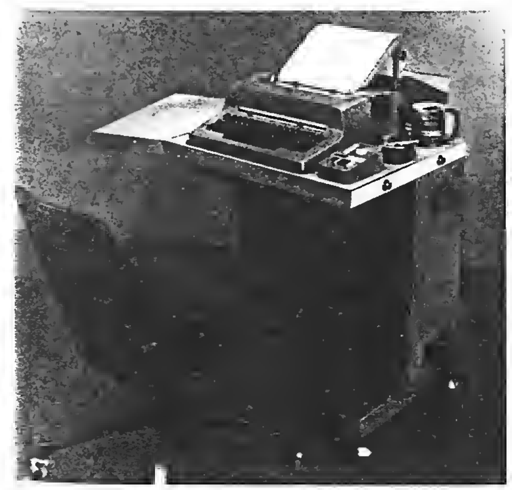


Figure 4. JOSS II terminal.

Figure 5. Chuck Baker at the JOSS II terminal.

The JOSS I "sound" seemed like a quiet doorbell chime to Chuck Baker, who considered the options for JOSS II. "A loud klaxon speaker we tried sounded terrible," he remembers. After experimenting with frequencies and duration, the JOSS II group picked a modest "beep" accomplished by a special beeper card that DEC built to drive a small speaker.

The terminal's simplicity was an important design specification. Appealing not only to the computer novice, the design also was an enticement to the casual user, who had little time to spend either learning the system initially or recalling what was learned perhaps weeks later. Complementing the language, the terminal played a major interactive role. Sight, sound, and touch each contributed. "Speed" as understood by the typical user did, too, and so did convenience and comfort.

The JOSS II terminal's workspace was a leaf that could be attached easily at either the right or left side, the wider part at the back or front as desired. With the leaf hung vertically instead of horizontally, the terminal could be wheeled through an office doorway to fit next to a desk or table (Figure 5). Attention was given to designing even the casters, which rolled equally well on tile or carpet (the latter no doubt for executive use), but did not skid out from under the typist during full-speed output.

A survey of potential users influenced the height of the terminal. The top was 29 inches above the floor, allowing for 24½ inches of knee room and an additional 1½ inches above the terminal top for comfortable use of the space bar. Once located, the terminal was simple to connect, by one cord to a power outlet and by the other to a JOSS outlet (if the room was so equipped) that led to the PDP-6 in the basement.

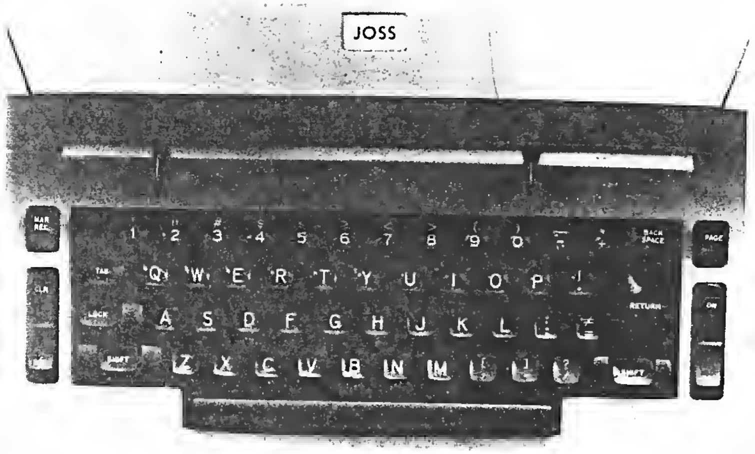


Figure 6. JOSS II keyboard of modified IBM 731 Selectric typewriter.

Cliff Shaw felt that the choice of character set and key positions for any on-line keyboard input device was not to be taken lightly. Chuck Baker echoed Cliff's concern for detail in noting the care with which the JOSS II terminal design was pursued. Chuck wrote in 1967, "Nothing was taken for granted, and at each step of the design process all available alternatives were reevaluated and original decisions reexamined." Chuck also gave great credit to the JOSS I terminal design. Its use provided the experience to decide which features had to be retained at all costs, which were desirable but not essential, what new ones to incorporate, and what deficiencies to overcome.

The JOSS I keyboard characters and their arrangement were preserved in JOSS II, differing from the standard typewriter set in only nine characters. Most retained their customary positions. Certain redundant ones were eliminated in favor of mathematical symbols and groupers, and there were adequate substitutes for eliminated business characters that might be used in text typing (Figure 6).

Since numeric expressions were to be typed in a line, an explicit sign for exponentiation was selected—a five-pointed, slightly elevated asterisk in upper case. The plus, minus (hyphen), multiplied-by (centered dot), divided-by (slash), and equals signs were all in lower case, the less frequently used inequalities in upper case. The duplicate comma and period in upper case were replaced by a pair of brackets to supplement the parentheses for improving readability in mathematical expressions. The vertical absolute value bar also served this purpose. Despite the few nonstandard positions among the 88 symbols, the touch typist soon adjusted, and the hunt-and-peck typist had no trouble following the rational layout.

Conventional rules of punctuation and capitalization apply in JOSS. Four symbols—the space sign (#), the asterisk, the underscore, and the dollar sign—have special uses, however. When the user catches a typing error before having pressed the Return key, he or she may backspace and type over with the correct char-

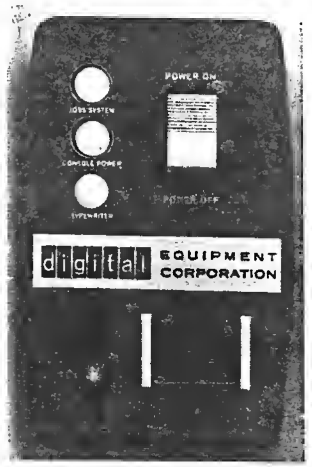


Figure 7. JOSS II control box.

Table 1. JOSS I Control Box

Switches		
	Power	On, off
	On	Connect terminal to JOSS
	Off	Disconnect terminal from JOSS
	Ready	Reactivate the typewriter after paper resupply
	In	Request control of typewriter for input
	Out	Relinquish control of typewriter for output
Lights		
	Power	On, off
	Enable	JOSS service is available
	Ready	Output is acceptable at the typewriter
	Red	JOSS controls the typewriter
	Green	User controls the typewriter
	In Request	User has depressed In switch but JOSS hasn't responded yet
	Out Request	JOSS has an administrative message for the user

acters, typing the space sign as a strikeover character where a blank should be. (Note that the space bar and backspace key only position a character, never enter one, when a line is initially typed into the line buffer.)

The asterisk typed at either end of a line causes JOSS to ignore the line when the Return key is pressed, and so is useful whether for correcting an error or for annotating the session. The underscore typed in a "Form" statement indicates fixed-point positions for a data field. The period is a floating-point position indicator for scientific notation. In JOSS I and II, the dollar sign carried the value of the current line number on the page. In JOSS III, it equals the number of lines remaining from a total of 54. So among its uses, the dollar sign can appear within an expression in a "Type" command to control print format.

Another aid to formatting JOSS output was the paging mechanism, which existed in JOSS I and was added to the Selectric typewriters used in JOSS II terminals. The paging operation moved standard-size fanfold paper over the page fold in a single motion; it was activated either by the Page key on the typewriter or by the Page command.

The JOSS I control box (Table 1) was slightly more complicated than the one for JOSS II (Table 2), but was physically almost as small and unforbidding as its successor.

For JOSS II, the six switches were reduced to two, the seven separate lights became five, and one light was combined with one of the two switches (Figure 7).

The JOSS I Ready switch was a Ready-Hold switch with two control positions and a spring-loaded neutral position. This would signal "hold up output" or "resume output" in case of paper jams or telephone calls. The Ready switch was retained in JOSS II, but was moved from the control box to the (unused) typewriter Power On-Off switch.

Pressing the JOSS II Interrupt button caused it to be backlighted in yellow to indicate that JOSS had received the user's request for control of the typewriter. The yellow light went out as soon as the user's green light went on. The result of a user interrupt is as follows.

If JOSS is typing, it always finishes typing the current line and sometimes a few more. If occupied with

Table 2. JOSS II Control Box

Switches		
	Power	On, off
	Interrupt	User requests control of the typewriter
Lights		
	Console Power	Turn-on signal has been sent to JOSS
	JOSS System	Turn-on signal acknowledged and power sent to typewriter
	Typewriter	Typewriter ready for input or output
	Red	JOSS controls the typewriter
	Green	User controls the typewriter

obeying a command to store items in a JOSS file, JOSS ignores the interruption until filing is complete or aborted for some other reason. If calculation is in process, JOSS types its position in the stored program before returning control (e.g., "I'm at step 3.2."). Otherwise JOSS simply types "Revoked by interrupt." Most important, the status of the user's block is exactly as it would have been if an interrupted command had never been started (except for the possibility that the line counter has been advanced).

The fundamental influence on terminal design was the line-at-a-time interactive mode of a conversational session. The user had complete control of the typewriter while composing an input line. Once the Return key had been depressed, JOSS either retained control of the typewriter—for the duration of the commanded task and perhaps outputting one or more complete lines—or immediately returned control to the user. The terminal was therefore always in one of two states, Red or Green.

In JOSS I and II, the Red state produced printed lines in black, and the Green state printed lines in green. In JOSS III, all printing is in black with the first character of a user-typed line underscored. The Red or Green state of the JOSS III terminal is always in effect, however, just as in JOSS I and II.

The user and JOSS needed to be aware at any instant which state was in effect. So the red and green lights furnished a continual feedback while the user was on line. The two-color typewriter ribbon left a permanent and unambiguous record. Further, when JOSS was in control, the keyboard locked. When the user received control, the keyboard unlocked and a gong (JOSS I) or beep (JOSS II) sounded. The gentle tone was a convenient reminder of control, which permitted the typist to concentrate on text or keyboard instead of flashing lights.

As for speed of response, JOSS I reacted to simple requests under a typical time-sharing load in a fraction of a second and rarely in as long as 3 seconds. Cliff Shaw observed that skilled typists could maintain impressive rates of interaction.

First priority for JOSS's attention went to servicing console signals—carriage return, page, on, off, in, out, and end-of-transmission. Second priority went to users with output-limited tasks that had been set aside until the typewriter had almost caught up. Third priority was given to users with unfinished computational tasks. Priorities, of course, change dynamically for an individual user during a session.

Joe Smith explained the design rationale in 1967. Short, direct action—editing a program or data, requests for "modest" computation—should be completed well before the typewriter carrier physically returned to its rest point. Most user-typed lines resulted in such short actions, so carrier return and page signals were assigned a high priority.

Output-limited tasks should produce a rhythmic pattern of typing without stuttering, so end-of-transmission signals had a high priority, too. Another top-rated usage was a short burst of processing followed by a long period when no processing was requested, as with users interrupting JOSS, or turning on or off. The end result for the JOSS user was a feeling that he or she generally had control of the typewriter when ready to use it. For the typical JOSS user, this was fast enough.

5. The JOSS User

In his 1967 Rand Report on console usage, Ed Bryan described the characteristics of a typical JOSS session. Among them:

Time at the console	46 minutes
Compute time	2.6 minutes
Input	82 lines
Mean program size	650 words (50% less than 200 words, 10% more than 2000 words)

During a typical 46-minute session, 15,000 JOSS statements were executed and 68,000 arithmetic operations were performed.

Ed made his summary after a year of JOSS operation on the PDP-6 computer, analyzing statistics accumulated by the JOSS monitor on usage by over 700 individuals. There were 400 different users each month, generating over 200 sessions each day and averaging 220 hours per month. During a prime shift, an average of 18 simultaneous users were recorded, with peaks of 25. Typical response occurred in less than carrier-return time. The mean task turnaround time, including all output and computation for satisfying a single-line request, was 10 seconds.

Ed felt it important to note that JOSS user requests are substantially different from those made on general-purpose, on-line, time-shared systems. In JOSS, there are a relatively large number of requests for short amounts of computing and a relatively small number of requests for a large amount of computing. Yet the amount of computing is by no means trivial, as can be seen from the number of statements and arithmetic operations performed.

Ed's 1966 report of JOSS usage concluded that a very fast response could be given to typical user requests. He attributed the response to a simple I/O-based priority scheduling coupled to a simple round-robin scheduling of compute-bound users. He felt the design goals were accomplished by providing a single easy-to-use language applicable to a wide class of problems *and* by strictly limiting program size and I/O speed (typewriter only, no tapes).

Ed also differentiated among the distinct types of usage: desk calculator, interactive problem-solving, production computing, output-limited computing, and multiconsole interactive games. He thought that future usage statistics might be more appropriately reported by types.

In designing the monitor, Ed took advantage of the system's interpretive software to record in detail the number and timing of statements, arithmetic operations, and user interactions. Such activity required at worst less than 1 percent of computing time. But in August 1972, data came directly from the users. That year, a survey questionnaire was circulated to gather information on which to base decisions about future JOSS-type service at Rand.

Analysis of the survey response (by Juan Robertson and Fred Blackwell) created a profile of users not only of JOSS but also of the IBM 360/65, especially those who used that system interactively. While the 360/65 batch service was rated very important, so also was JOSS, for such reasons as convenience, simplicity, fast response time, low cost, 24-hour availability, and the conversational nature of the system.

Scientific calculations accounted for two-fifths of JOSS use, with simulation and statistical programs together contributing another third. The dominant job classifications of respondents were engineer, mathematician, operations research analyst, programmer analyst, and economist. Physical scientists, research aides, administrators, social scientists, and an assortment of other specialists also participated.

Of the total group questioned, there seemed to be a division between the relatively unsophisticated computer users (JOSS) and the sophisticated users (non-JOSS, especially users of large computers via terminals). Despite the increasing availability (in 1972) of versatile interactive systems, their relative complexity made a nucleus of JOSS users unwilling to switch.

The question of how JOSS work would be affected if access were delayed by system malfunction proved interesting. If the delay were 1 hour or even 1 day, there would be no serious effect, respondents thought. The feeling was quite different if the delay were a week or as much as a month. Only 30 percent indicated they had effective alternatives, which ranged from paper and pencil or slide rules to desk calculators or even the 360/65 in some instances. Clearly the dependence on JOSS was heavy.

The survey pointed up another peculiarity of the typical JOSS user—reliance on a trial-and-error method of learning the system. Cliff Shaw wrote in May 1965: "an hour's demonstration at a typewriter console is still the most effective introduction." Cliff rejected learning by means of a formal description of the language, which he felt would be at least as difficult to learn as the language itself. What the new user could not discover personally could be asked of a more experienced user. One was available in every Rand department to assist the novice.

Cliff recalled an observation made near the end of the JOSS I initial test period. A senior staff member, not one of the JOSS test volunteers, was seen using the system before any training program had been set up. It seems that "he was taught by another unauthorized senior staff member who was taught by still another unauthorized senior staff member who learned by looking over the shoulder of an authorized user."

By the time JOSS II was operational, an abundance of documentation had been published, some of it intended for user reference. Ed Bryan and Joe Smith (1967) coauthored a brief reference summary in three handy formats: book-size, pocket-size, and poster-size. Bryan and user Ed Paxson (1967) together produced a small ring-binder of annotated examples recording a mathematician's efforts to learn the system. Shirley Marks and G. W. Armerding (1967) wrote a self-teaching primer, which introduced the learner to the system while seated at a terminal. All these were printed in the conversational green and black of a JOSS session. Larry Clark (1975) prepared a comprehensive reference manual for JOSS III users, with emphasis on examples.

The 1973 user survey by Robertson and Blackwell rated self-paced documentation developed early in the system's life as an important factor in acquiring user skills. There seemed to be implications for interactive systems generally. In addition to the documentation for the learner, there were the regular communications set up for users both at Rand and at sites across the country.

A monthly newsletter, published for JOSS II users from November 1967 through June 1971, touched on topics as varied as random-number generation and hardware malfunction. The latter was featured among the most dramatic memories of those days. The news began with the JOSS Flash of November 26, 1968, "Items stored in files since November 7th are lost. . . ."

A more immediate means of communication with JOSS I and JOSS II users was the administrative message, which could appear in the heading when JOSS began a new page for a particular user. The heading normally contained the time, date, terminal number, log-on identification, and current page number. There was room to the right of this information for a brief

message, which might simply be that maintenance shutdown would occur at 2100. Shortly before such routine shutdown or for a high-priority message, the PDP-6 operator would initialize a "beep" to all consoles. This signal was sent once per second for the first 5 seconds of each minute, for as long as the operator wished. The beep not only alerted users to press the Page key and read the heading message, but also frequently startled nonusers as it resounded through the Rand halls. For longer text than the space allowed, the heading message could suggest that the user recall a "public" file known as the JOSS Mailbox.

The newsletter and the administrative heading message created a bridge between the nonprogrammer user and the hardware-software setting of the Rand basement. Most often what the typical user needed was merely advance notice of maintenance and occasionally a brief explanation of system trouble that might affect ongoing work. Even the most adept of JOSS users seldom ventured into that never-never land behind their terminals, very likely because they identified JOSS with the terminal itself. Anything that affected response time *was* of interest, though, especially excessive use of files. So it was necessary from time to time to divulge some details of the cause in order to describe the effect.

JOSS users today live within the world of an IBM operating system that caters to other systems beside JOSS. That operating system can be "tuned" to improve JOSS response. In a typical 2-week period, about 20 Rand people and 12 Rand clients incur charges, although it is not clear whether all charges are for active usage. Solving small computation problems is still the principal application. Users are mainly those Rand staff members who calculated with JOSS in its early days plus those who have heard of it from the originals. There are no plans to improve or change JOSS or to modify its present level of support.

6. The JOSS Influence

In December 1972, the Joint Publications Research Service of Arlington, Virginia, published a description of the Russian AIST-O system developed at Novosibirsk. According to this English translation, an AIST-O user seated at a Siberian terminal could invoke any of several programs, one of which would greet him with "DZHOSS at your service. Name of problem:" The user could then create a program of "steps" and "parts" in a simple (presumably Russian-like) language. DZHOSS (an English transliteration of the Cyrillic name for the JOSS-like Russian program available in the AIST-O system) responded to syntactic errors with "What?"

Despite appearances, this "adaptation" of JOSS was based only on the elementary user documentation supplied after a Russian computer scientist had visited Rand. No credit was given to the source of DZHOSS, but then the similarity was a surface one. The JOSS software and complete documentation *were* provided to several computer organizations in the United States, however. The intent was that a JOSS conversion would rapidly spread the concept of simple interaction for the nonprogrammer.

The principal constraint placed on these organizations was in the use of the name JOSS, which required Rand's consent after a rigorous examination for equivalence to the PDP-6 model. No commercial version was ever approved to this extent. The result of such a sheltered existence has been limited fame and certainly no fortune (none was possible, of course, since Rand remains a nonprofit corporation and the JOSS project was sponsored by the Air Force).

Nevertheless, the JOSS touch can be seen as computing power has become available to an ever broader class of user. Over the years, many languages have been touted as "easy to learn and easy to use." Early derivatives of JOSS include CAL, CITRAN, ESI, ISIS, PIL/I, and TELCOMP. The modern computer terminal seems also to reflect the JOSS influence. Keyboards now resemble those of typewriters instead of the unfamiliar teletypewriter that served as handy input device in the early time-sharing days. Surely such user-oriented design features have enhanced the popularity of terminal systems in classrooms, banks, airline reservation offices, and brokerage firms.

The BASIC language, which was developed in 1964 at Dartmouth College by Kemeny and Kurtz, took a different course from that of its contemporary, JOSS. Standing for "Beginner's All Purpose Symbolic Instruction Code," BASIC was intended by its designers to introduce students to the computer. It was hoped that they would then be encouraged to progress to more powerful languages.

Although BASIC is considered both elementary and conversational, BASIC at Dartmouth differed from JOSS in an important aspect—its statements were compiled, not interpreted. Many of today's BASIC implementations are interpretive and some are even interactive, whereas Dartmouth versions have always been compiled. Thus Dartmouth BASIC required a full program, which could not be restarted from a point of interruption. BASIC also differs from JOSS in its proliferation, for today there are many variations for many computers, all called (basically) BASIC. Both languages are alike in having simplified access to the computer for many who might not otherwise have been served.

Visitors to Rand during the heyday of JOSS were also subjected to the missionary message of computing power for the computer novice. Students and foreign dignitaries alike sat at blue terminals, typed and mistyped, calculated and played games, and almost always overcame their initial reserve in an enjoyable session with the unseen computer. Even when they took a trip to view the power behind the terminal, they continued to identify with the system's personal response instead of the impersonal flashing of PDP-6 console lights.

Student visitors varied in sophistication from a university graduate class in mathematics to preschoolers who took turns holding down the JOSS typewriter Shift key. For all, the answer to "What happens if . . . ?" was "Let's try it and see." And they usually were eager to try. So as a teaching tool, JOSS was most useful in allowing students to experiment without worrying about the consequences. Not only could they discover the answer for themselves, the answer often led to another question. Most important, JOSS was always there ready to help under all conditions except a power outage or computer failure. Nothing the user could do (short of taking a hammer to the terminal) could kill JOSS or other users' programs, or even make the user's own program unreadable to JOSS.

Nonnumeric games like Hangman enabled the word-oriented visitor to participate, too, and to experience the personal response. Some even forgot they were conversing with a computer, referring to JOSS as "he" instead of "it." No official brainwashing was involved, or intended. Kids simply felt at ease and expressed themselves naturally. One young fan even sent JOSS a valentine card after a friendly visit.

Adult reaction was memorable, too. A delegation of Japanese industrialists smiled happily at the JOSS "Eh?" when this mysterious typed message was explained by their interpreter. A proper British gentleman from Whitehall began timidly to play a JOSS game of Blackjack, then with difficulty was persuaded to relinquish his computerized winnings in order to continue his tour of Rand. Unforgettable was the day an Italian television company came to shoot a film and referred to the system as "HOSS" in true Western-movie style.

In 1966, Ed Bryan put it all in perspective before a meeting of DECUS, the Digital Equipment Computer Users Society. Said Ed, "A list of the limitations of JOSS is perhaps more instructive than one of its capabilities." He then remarked that JOSS is poor at data retrieval, can't handle large data files, can't do large programs, is impractical for long-running programs, can't do high-volume input-output, and provides no generalized symbol manipulation.

Ed balanced these inadequacies against JOSS's limited design goals, which offered a simple tool for the occasional user. Generally such a user spends less time from problem inception to solution by means of a JOSS-like system than with less accessible types. What a user may have forgotten about rules since the last session can be brought back to mind by experimenting at the terminal. In this way, many problems that may not be worth the effort in another computing environment can be explored to advantage.

Stating it in less formal terms, a ninth-grader wrote in a class report following a Rand visit, "The JOSS computer wasn't just a tool one would work with, it was more like a friend who helps you with your work. He's also polite."

REFERENCES[5]

Baker, C. L. February 1967. *JOSS: Console Design.* Rand Corporation, RM-5218-PR.

Baker, C. L. 1981. JOSS—JOHNNIAC Open-Shop System. In R. L. Wexelblat (ed.), *History of Programming Languages*, New York, Academic Press, pp. 495–513.

Bryan, G. E. November 1966. *JOSS: Introduction to the System Implementation.* Rand Corporation, P-3486.

Bryan, G. E. August 1967. *JOSS: 20,000 Hours at the Console—A Statistical Summary.* Rand Corporation. RM-5359-PR.

Bryan, G. E., and E. W. Paxson. August 1967. *The JOSS Notebook.* Rand Corporation, RM-5367-PR.

Bryan, G. E., and J. W. Smith. August 1967. *JOSS Language.* Rand Corporation, RM-5377-PR.

Clark, R. L. January 1975. *The Rand Computation Center: JOSS Users' Reference Manual.* Rand Corporation, R-1555/9.

Greenwald, I. D. September 1966. *JOSS: Arithmetic and Function Evaluation Routines.* Rand Corporation, RM-5028-PR.

Joint Publications Research Service. December 18, 1972. *Card Deck, Software Instructions for AIST-O.* Springfield, Va., National Technical Information Service, JPRS 57790.

Kurtz, T. E. August 1978. BASIC. *SIGPLAN Notices 13*, 8, 103–118.

Marks, S. L. December 1971. *The JOSS Years: Reflections on an Experiment.* Rand Corporation, R-918.

Marks, S. L., and G. W. Armerding. August 1967. *The JOSS Primer.* Rand Corporation, RM-5220-PR.

Sackman, Harold. 1970. *Man-Computer Problem Solving.* Pennsauken, N. J., Auerbach Publishers.

Sammet, J. E. 1969. *Programming Languages: History and Fundamentals.* Englewood Cliffs, N. J., Prentice-Hall.

Shaw, J. C. August 1964. *JOSS: A Designer's View of an Experimental On-line Computing System.* Rand Corporation, P-2922.

[5] A complete list of JOSS documentation can be obtained from the Rand Corporation, 1700 Main Street, Santa Monica, CA 90406, Attn. Publications.

Shaw, J. C. May 1965. *JOSS: Experience with an Experimental Computing Service for Users at Remote Typewriter Consoles*. Rand Corporation, P-3149.

Smith, J. W. August 1967. *JOSS: Central Processing Routines*. Rand Corporation, RM-5270-PR.

Ware, W. H. March 1966. *JOHNNIAC Eulogy*. Rand Corporation, P-3313.

APPENDIX

The JOSS Cast of Characters
(Plus Three Generations of Users and Visitors)

Armer, P.	Paul Armer headed the Computer Sciences Department, within which JOSS was developed
Armerding, G. W.	George Armerding headed the JOSS II operational staff when the design/implementation group's work was completed
Baker, C. L.	Chuck Baker headed the JOSS II design/implementation group, which also included Bryan, Greenwald, and J. Smith
Bernstein, M. I.	Mort Bernstein was charged with developing a software system for the JOSS I hardware in case the effort under Shaw failed to produce a workable software system
Bryan, G. E.	Ed Bryan was a member of the JOSS II design/implementation group
Clark, R. L.	Larry Clark was a member of the JOSS II operational staff and installed JOSS III on an IBM Model 370/158
Cooper, D.	Donna Cooper assisted Clark in installing JOSS III
Cutler, L.	Leola Cutler assisted Shaw in writing JOSS software for the JOHNNIAC
Davis, M. R.	Mal Davis managed (jointly with Ellis) the JOSS I communications construction
Ellis, T. O.	Tom Ellis managed (jointly with Davis) the JOSS I communications construction
Greenwald, I. D.	Irwin Greenwald was a member of the JOSS II design/implementation group
Gross, O. A.	Oliver Gross was a pioneer user of JOSS I and provided JOSS II designers with valued feedback
Gruenberger, F. J.	Fred Gruenberger demonstrated JOSS for visitors and used the system in school classes
Lind, M.	Mary Lind assisted Shaw in writing JOSS software for the JOHNNIAC
Lucero, A.	Art Lucero maintained the JOSS II hardware
Marks, S. L.	Shirley Marks attended to JOSS II users and visitors
McClenon, P.	Paul McClenon promoted JOSS use in Washington, D.C.
Newell, A.	Al Newell, Cliff Shaw, and Herb Simon had a joint research project on artificial intelligence of which JOSS was a subproject
Paxson, E.	Ed Paxson did some of the earliest teleconferencing research using JOSS
Shapiro, N. Z.	Norm Shapiro was an early user who provided valued feedback to JOSS I and II designers
Shaw, J. C.	Cliff Shaw was the father of JOSS
Sibley, W. L.	Bill Sibley was an early user who frequently demonstrated JOSS I to visitors
Simon, H. A.	See Newell
Smith, A. C.	Art Smith was an early user who provided valued feedback to JOSS I and II designers
Smith, J. W.	Joe Smith was a member of the JOSS II design/implementation group
Uncapher, K. W.	Keith Uncapher headed the group of engineers involved in the construction of JOSS and was a staunch JOSS advocate
Ware, W. H.	Willis Ware was associate head of the Computer Sciences Department when JOSS development began
Williams, J. D.	John Williams was head of the Mathematics Department and a pioneer JOSS user and supporter

Meetings in Retrospect

J. G. Brainerd on the ENIAC

On June 25, 1981, John Grist Brainerd gave the Sixth Pioneer Computer Lecture at the Digital Computer Museum in Marlboro, Massachusetts. Brainerd was in charge of the ENIAC development at the Moore School of Electrical Engineering at the University of Pennsylvania in the early 1940s. The following comments about the lecture were prepared with Brainerd's assistance.

The Moore School of Electrical Engineering was endowed independently as part of the University of Pennsylvania in 1923. Harold Pender became dean and remained so during the war period. He assembled around him a young faculty who were both innovative and dynamic. Five of them, Irven Travis, Knox McIlwain, Charles Weyl, Carl Chambers, and Brainerd himself, were all interested in physical problems involving nonlinear differential equations. These are rarely soluble analytically, and in those days their numerical solution presented great difficulties. Travis had the idea that they should try to build a differential analyzer, modeled after the small one built by Vannevar Bush at MIT. In order to obtain assistance from the Civil Works Administration (a governmental agency of those depression years) it was necessary for the project to be endorsed by another governmental agency. Through a Moore School graduate who worked there, they approached the Ballistic Research Laboratory (BRL) at Aberdeen Proving Ground, Maryland. BRL agreed to endorse the project, which then went ahead. Bush cooperated magnificently, as he always did, and they were able to go far beyond their original dreams and build a 10-integrator differential analyzer; this was probably the largest computing machine in the world at the time.

The project just mentioned gave the Moore School some experience with the handling of a large government-sponsored project. Not only was experience with such projects extremely rare in university departments in those years, but there was an extensive prejudice against them in the liberal arts faculties which tended to dominate universities. This feeling recurred during the Vietnam War when, as Brainerd put it, "they took some severe cracks at us." Brainerd had also had some experience with the network analyzer at Westinghouse during a year's leave of absence. This was a large-scale analog device, primarily used for studying power network stability. Travis had been a consultant to General Electric and in that capacity had looked into the possibility of ganging desk calculators together to make a computer. He had reported negatively on this idea, but it led him to an interest in computing devices, and he and Brainerd often discussed the possibility of building an electronic machine.

By 1938–1940 relations with BRL were becoming close. The differential analyzer was being used for ballistics calculations, and in addition the Moore School had been asked to recruit and administer a group of 100 women to operate desk machines. This was to supplement a similar group at Aberdeen. On the engineering side, the Moore School accepted a number of small contracts, including one from the Radiation Laboratory at MIT on the timing of radar signals using mercury delay lines and ring counters. For this purpose Brainerd assembled a team consisting of J. Presper Eckert, T. Kite Sharpless, Joseph Chedaker, and Vincent Porter (who soon resigned).

The result of the developments just described was to establish in the Moore School a receptive climate for further developments in the computing area. In particular, it became clear that it would not be possible to keep up with the computing needs of BRL by recruiting more and more women who could operate desk calculating machines and also understand the background mathematics.

Two people, who in their different ways were to play major parts in the ENIAC project, were involved in its beginnings. One was John Mauchly, who had had some computing experience and was interested in weather problems and also to some extent in statistics. The other was Herman Goldstine, an assistant professor of mathematics at the University of Michigan, who was serving as a lieutenant in the U.S. Army. He was attached to BRL where there was a mathematical tradition going back to World War I, and he had been posted to the Moore School as liaison officer. Mauchly did not at that time know very much about electronics, but his thinking was of a high order, and in August 1942 he wrote a memorandum in which he proposed the construction of an electronic computing machine. Brainerd related how this memorandum was lost and later reconstituted from a secretary's notes. When he saw it in March of 1943 he had no hesitation in